Modeling Dendritic Structures for Artistic Effects

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By

Jeremy Long
Department of Computer Science
University of Saskatchewan
jsl847@mail.usask.ca

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

Dendritic or branching structures are commonly seen in natural phenomena such as lightning, cracking and vegetal growth. They are also often used for artistic or decorative purposes, ranging from ornamentation to decorative ceramics. Existing procedural methods for modeling these structures remain very limited in terms of control and flexibility. As a result, these objects tend to be modeled individually, which is a painstaking and costly process.

We present a new procedural method for modeling dendritic structures based on a path planning approach. Our method includes the implementation of a partial non-scalar distance metric that gives us effective and flexible control handles over the evolving dendritic structure. These control handles are demonstrated by guiding the growth of dendritic structures using input images, allowing us to create a form of stylistic dendritic halftoning and to embed hidden images in dendritic trees to create pareidolia effects. These applications demonstrate the vast diversity of structures that can easily be modeled by our process – a flexibility that existing methods definitely lack. We also demonstrate the application of the partial non-scalar distance metric to the context of texture synthesis from example, and show how it holds promise for many other contexts.

Acknowledgements

The author would like to thank Professor David Mould for his guidance throughout the past two years.

The lightning rendering engine that was applied to create figure 3.33 was developed by Ling Xu, and we thank her feeding our input branches through this engine.

Our spline implementation is based on the GoTools core library, developed by the geometry group at SINTEF ICT. This library facilitated our use of splines for our vine halftoning.

We would also like to thank Andrew Nealen and Marc Alexa for releasing their Hybrid Texture Synthesis code for use by the research community. This Matlab code greatly assisted with the rapid prototyping of our Improved Image Quilting.

Finally, we would like to thank Professor Eric Neufeld for allowing his likeness to be dendrified as part of our results, and the thesis committee for their comments and suggestions.

Contents

List of Figures

Chapter 1

Introduction

Many objects that appear in nature display a dendritic or branching structure, such as lightning, frost, cracks, and many forms of vegetation. Dendritic structures have also long been used for ornamental and decorative purposes. Illuminations are an example of a traditional medium that employs vines and other floral material to create a compelling artistic effect, as shown by the manuscript and ceramics shown in figure 1.1. In addition to decorating letters and ceramics, these branches are also sometimes used to make aesthetic patterns or to convey an image [39, 60].

Figure 1.1: An illuminated manuscript featuring dendritic structures (left) and ceramics featuring floral ornamentation (right).

Effectively modeling dendritic structures remains a challenge for computer graphics researchers. The results generated by existing procedural methods, while at times compelling, are notoriously difficult to control and expensive to compute. A procedural method can be defined as an algorithm that specifies some facet of a computer–generated model [13]. In general, procedural approaches are intended to minimize user interaction. Given the weaknesses of existing procedural techniques

for modeling dendritic structures described above, many companies rely on computer artists to create these sorts of objects individually, which is a painstaking and costly process.

This thesis addresses the problem stated above by presenting a new procedural technique for modeling dendritic structures using path planning. This yields more complete control over the final result, and a significant decrease in the computational time required for the process. The use of a non-scalar distance metric for the path planning algorithm allows for the creation of better space-filling branches for the resulting structure, and further enhances the control over the evolving dendritic structure.

This new method presents great promise in many different contexts, ranging from the modeling of natural phenomenon to non-photorealistic effects. We will specifically apply this technique to the domain of artistic halftoning, where an input image can be embedded into a dendritic structure and then rendered using different sorts of metaphors. We also show how our modeling method can be modified in order to generate trees whose branches can display images, creating a sort of pareidolia or hidden image effect. We will also discuss how the partial non-scalar distance metric that we use can be applied to other contexts, such as non-parametric texture synthesis from example [30], though we believe it holds potential in many other areas as well, including Wang tiles [53, 54], line art, ceramics, and decorative mazes [39, 61], to name only a few. In particular, we believe our method is promising for artistic effects because we believe it is easy to control, which is necessary for media where exacting precision is often required.

The main contribution of this work is the use of the partial non-scalar distance metric in the path planning algorithm for generating dendritic structures. These dendrites can be deployed to create compelling artistic effects. Throughout this work, we demonstrate how the control characteristics of this method are conducive to modeling visually appealing dendritic structures. Furthermore, our method is extremely versatile, and can generate many different sorts of dendritic structures without significant changes to the modeling process. A further contribution is the demonstration that the partial non-scalar distance metric can be employed in other areas where path planning is used, such as non-parametric texture synthesis from example [30].

1.1 Overview

Chapter two of this thesis will begin by discussing related and previous work in the relevant areas of dendritic structures, path planning, halftoning, tree modeling, and the minimum error boundary cut. With this context established, we will then describe several contributions of our work in the chapters to follow. These contributions all stem from the same procedural path planning approach that we use to model our dendritic structures.

Chapter three focuses on methods for modeling dendritic structures. We start by describ-

ing some of the existing procedural methods for generating dendritic structures in greater detail. We examine four of the more common procedural methods based on their control characteristics, aesthetics and computational efficiency, and in so doing we show that they are not suitable for procedurally generating dendritic structures where exacting precision and control is expected, such as is the case when modeling dendritic structures for artistic effects.

Chapter three continues by describing our own procedural framework for modeling dendritic structures based on a path planning approach. We also define the partial non-scalar distance metric that grants our method very favorable control characteristics. We then apply this method to stylistic halftoning and render the results using a vegetal metaphor.

The final section of chapter three describes how our modeling process can be easily modified to model skeletal trees. We show our algorithm can be used iteratively to form multi-octave trees with different sorts of branch characteristics. By guiding the growth of the tree branches using an input image, we can create a form of pareidolia effect where some of the features in the source image can be inferred from the structure of the branches.

Chapter four describes another use for the partial non-scalar distance metric. The control properties that make this metric well suited for artistic applications also make it effective as a means of finding the minimum error boundary cut, which is used in applications such as non-parametric texture synthesis from example. We show how this alternate distance metric performs better than existing methods that rely on the cumulative distance metric.

In the final chapter of this thesis, we will formulate conclusions and reiterate the main contributions of our work. We will also discuss some ideas for future work that might improve the results we were able to achieve, in addition to suggesting some other contexts in which the partial non-scalar distance metric might prove fruitful.

Chapter 2

Related Work

2.1 Dendritic Structures

Initial work on the modeling of dendritic structures drew much inspiration from research into fractals. Fractals, or highly recursive structures, have often been used to model complex natural phenomena, such as trees and snowflakes [11]. Because of their success in this area, many attempts were made to adapt fractal generation methods for use in modeling dendritic structures. This led to the use of Lindenmeyer-Systems (L-Systems) [43, 44].

An L-System can be described as a replacement grammar that evolves initial axioms using a set of production rules over a number of discrete derivation steps [43]. This method has been used to produce highly recursive structures, such as can be created using LOGO or Turtle Graphics [1]. However, early L-Systems relied only on endogenous control – the growth of the structure was solely dependent on the current state of the structure itself, not on any part of the environment. This greatly limits the amount of control that can be exerted on the evolving objects. Furthermore, this control mechanism is extremely unintuitive, as small changes within the production rules can lead to extremely unpredictable results. Both of these reasons make L-Systems impractical for our purposes, as creating detailed artistic effects requires precise control of the evolving structure.

These issues were addressed in part by the introduction of context-sensitive L-Systems, often referred to as Open L-Systems [31]. In this approach, the spatial position of the branches can be used in calculating the next production rule. To accommodate this, the current string is interpreted at each iteration. This offers more control over the evolution of the structure, since it can now draw influence from spatial coordinates. However, the control handles present in Open L-Systems remain unintuitive. While the production rules offer more flexibility than in regular L-Systems, they are still the only control handle for the growth of the object, and it is difficult to predict the results that will emerge from changes to the control handles.

The growth of dendritic structures has also been studied within the context of ornamentation. Wong et al. [60] introduced their own grammar for creating ornamentation. Though similar in concept to L-systems in that it involves elements and production rules, this method features the serial processing of rules as opposed to the parallel processing seen in L-systems. However, this

approach is aimed only at floral ornamentation, and concentrates on filling a planar region. We find that our method allows us to model dendritic structures for a wide variety of effects aside from just ornamentation, and can be used to fill a region of arbitrary size rather than being restricted to a plane.

Another direction that has been taken involves the use of physically based processes to generate dendritic structures [3]. These processes range from fluid flow models to the aggregation of dust particles, and have shown much promise. Some of these techniques have been used in combination with the simulation of natural processes to model the complex formation of ice crystals [26, 25], the growth of lichen [12] and for animating lightning [27]. These approaches tend to function best when they are intended to simulate the natural process on which they are based. However, they tend to suffer from a lack of control. Furthermore, these methods do not tend to be very flexible, and it is difficult to use them to model objects aside from the ones for which they are intended. We introduce some of these methods here, and they will be discussed more thoroughly at the beginning of chapter three.

One of the first of these natural processes proposed for growing dendritic structures was the Eden Model [6, 3], which allows the object to grow randomly from some seed point. This technique later gave birth to the Modified Eden Model, which grows according to some probability distribution that is specified across the lattice [3]. Although the latter of these methods does present some manner of controlling the evolving structure based on the use of a probability distribution, it is still very difficult to predict the exact manner in which the structure will respond to changes in the control handle and as such it does not easily produce compelling results.

Another natural process that has been simulated is invasion percolation, which is based on fluid flows [50]. It assumes that the fluid will always spread along the path of least resistance. The resulting structures closely resemble river systems. However, this method has no semblance of global control. This means that although we can control the branches on a local level, we have no way of directing the branches towards a specific destination, or of specifying the total dimensions of the dendrite. This makes invasion percolation unsuitable for our purposes.

One of the most common methods of generating dendritic structures is DLA (diffusion-limited aggregation), which simulates the progress of dust particles floating through the air [5, 6]. Particles are released a long distance from the structure and allowed to perform a random walk until they reach the structure [59], where they attach themselves to it. This method can produce very compelling results that greatly resemble branching structures one might expect to see in the real world, as shown in figure 3.7. However, DLA is extremely costly to compute and difficult to control.

Path planning is the process of finding a least cost traversal through a weighted graph [58, 10]. It is a well defined problem in computer science, and has often been used in the contexts of artificial intelligence and computational geometry [18]. By originating all the paths from a single seed point

in the graph and pathing to an arbitrary array of endpoints, we are able to obtain a structure that possesses dendritic properties such that the branches of the structure do not cross over each other. The result is also fast to compute, and more importantly, offers very powerful control mechanisms in the forms of the edge costs and the placement of the endpoints. In other words, the effects produced by changes to the control handles can easily be predicted.

The distance metric that we use for generating our dendritic structures was first introduced by Pai and Reissell [38] as a means of improving path planning for robots. Their non-scalar distance metric attempts to model the roughness of terrain, under the premise that robots are safer by avoiding peak edge costs, even if it means taking an alternate route with a higher total cost. This results in longer, more winding paths. As we discuss later, this property allows the non-scalar paths to fill space more effectively than those generated by the conventional distance metric. The control characteristics demonstrates by this method make the non-scalar distance metric ideal for modeling dendritic structures, but also seem useful for other applications in which path planning has been employed, such as texture synthesis [30], and Wang tiles [53, 54, 8].

2.2 Halftoning

The method that we have developed for modeling dendritic structures shows great potential for achieving artistic halftoning effects. Halftoning has been a topic of great interest in the computer graphics community. In its simplest form, halftoning can be described as a process through which the continuous color resolution of an image is discretized while still maintaining as much visual coherence as possible. This often results in a black and white image, which can subsequently be used for stylistic or printing purposes [64].

While conventional halftoning aims to avoid halftoning artifacts where the distribution of halftoning primitives produces a spurious pattern, non-photorealistic halftoning attempts to embed specific primitives, such as strokes or stipples, within the output in order to achieve a certain artistic purpose [52, 17]. This has included the simulation of various traditional media, such as painterly rendering [49], pen and ink style [47, 57] or stippling [48]. We attempt to model the sorts of dendritic structures that have traditionally been used for many decorative applications, such as vines and tree branches.

The techniques described above employ different types of halftoning primitives in order to simulate their mediums of choice. Stippling involves the use of small, single-colored dots as the halftoning primitives [48], while strokes of different size and orientation are employed for painterly rendering and pen and ink [49, 57]. Some approaches view a primitive as a group of strokes, as seen in hatching [42]. In our case, we use the paths generated through the algorithm described in this thesis as our primitives, yielding a dendritic structure that resembles a stylistically halftoned

version of the original input image, as shown in figure 2.1.

Figure 2.1: Images that have been dendritically halftoned.

The primary goal of artistic halftoning is to achieve a compelling balance between reproducing the artistic style in question and preserving salient features in the input image. Preserving features such as gradient edges has traditionally proven difficult. Some methods circumvent this complication by allowing users to highlight important features in the input [47]. Streit and Buchanan proposed the notion of a generalized importance map that could be used to place primitives procedurally where they are most needed to convey the essence of the input image [51]. We find that this concept can easily be customized for our approach in order to accurately capture important image features, including gradient edges, without user interaction.

The notion of multiresolution curves, first devised by Finkelstein and Salesin [16], shows potential for improving the results generated by halftoning approaches that use strokes as their primitives. This curve representation is useful for smoothing or perturbing the results as much as is appropriate for the given medium. We employ a reduced version of multiresolution curves, which allows us to arbitrarily smooth our halftoning dendritic structure.

There is considerable precedent in using images to guide the growth of various unusual yet artistic structures. Mould [33] used an input image to guide the evolution of crack patterns. Pedersen and Singh [39] and Xu and Kaplan [61] shape labyrinth structures around a source image. Wong et al. [60] create floral ornamentation intended to fill a planar region. Our approach uses a different method for growing the structure, but we share the goal of representing an arbitrary input image using an unusual set of primitives.

This is not the first time that path planning has been used to search for gradient edges.

Mortensen and Barrett employed path planning to locate features in an image using edge boundaries [32]. Although this is similar to our method, the classical cumulative distance metric employed in their approach tends to take short cuts over high cost areas in order to decrease the total length of the path, making it less effective for halftoning and other artistic effects. We have found an alternate non-scalar distance metric that does not suffer from this weakness.

2.3 Image-Guided Tree Modeling

The procedural generation of trees has been addressed several times by computer graphics researchers. Most of the approaches that have been taken are intended to model photorealistic trees. Our method for modeling dendritic structures shows promise for growing non-photorealistic trees. One of our main interests lies in creating pareidolia effects, where images (usually faces) are seen within a natural structure such as a tree. The framework that we have developed seems conducive to creating these sorts of effects.

Several forms of L-Systems have been used to create biologically faithful and visually realistic models of plants, including trees [60, 43, 44, 40]. More recent work on this area tends to focus on simulating the interaction between the plants and their environment in order to use information from these natural processes during the growth of the plants, and their placement in a scene [40, 45]. Several approaches have also been developed for interacting with and manipulating grammar-based models, though they remain difficult to control [65, 22].

Some graphics researchers have been more interested in conveying the appearance of trees without worrying about the natural processes that form them. A few of these approaches have created tree models based on geometric considerations [55] or particle systems [46]. These methods can produce results that resemble real trees reasonably well, at the expense of control and flexibility. Because these methods are designed to model individual types of trees, they require almost completely new sets of instructions in order to create different types of trees or even irregular characteristics within the same type of tree. This lack of control makes these methods unsuitable for containing hidden images.

Our approach is not intended to model the natural processes involved with vegetal growth, nor is it expected to create photorealistic results. We are more interested in achieving the sorts of artistic effects that can be seen in landscape painting, where highly stylized trees and vegetation are often present. Most computational approaches to painterly rendering tend to operate as image filters [9, 19]. They take in a source image and transform the pixels into paint strokes. Some methods have gone to great lengths to simulate the physical properties of the fluids used in painting [4, 7, 37].

Our challenge is a little different. We want to transform an input image of any sort into a painted tree representation. This feat is difficult to achieve using traditional methods that rely on pixel

transformations to create paint strokes. Tree branches have dendritic properties, and as such we can consider this a special case of dendrites that we can model using our algorithm. Furthermore, our algorithm allows us to guide the growth of the tree models using an input image, allowing us to create pareidolia type effects.

2.4 The Minimum Error Boundary Cut

Chapter four of this thesis will show that our partial non-scalar distance metric is useful in other applications where path planning is employed. One example that we have examined is the minimum error boundary cut. This section introduces non-parametric texture synthesis from example and provides some background for our investigation.

Non-parametric texture synthesis from example is the process of taking an input texture chip and using it as a basis for generating an arbitrary quantity of similar texture, without obvious repetition. One way this can be accomplished is by copying parts (pixels or patches) from the input texture and pasting them together as an output texture [15, 56, 2, 63, 23]. Our contribution falls within the domain of patch-based synthesis. The minimum error boundary cut was first introduced as a means of reducing error in non-parametric texture synthesis from example, though it has since been used in other contexts such as Wang tiles [14, 53, 54, 8].

Patch–based texture synthesis approaches attempt to synthesize whole patches of pixels at once, usually by choosing to copy from the input texture chip the patch that best overlaps with the part of the output texture that has already been generated. Implementations of this method [62, 14, 29, 35, 36, 41] tend to preserve features in the input texture, assuming the chosen patch size is large enough to encompass them. However, the naive patch-based approach can leave discontinuities along the seams of the patches, unless some method is applied to avoid or repair them. Considerable research has been devoted to discovering the best manner to accomplish this.

Several methods for removing the seams on the boundaries of patches have been investigated, including alpha blending [29], the shaping of irregular patches [14], and post-process pixel re-synthesis methods [35, 36]. Our contribution falls into the former category, so we shall describe some of the existing methods for creating irregularly shaped patches in detail.

One of the first methods for patch–based synthesis was Liang et al.'s feathering approach [29]. This technique uses alpha blending across the overlap region to mix the new patch with the results synthesized in previous iterations. While feathering is effective for preserving global structure, it comes at the cost of significant blurring along the edges of the patches. This defect can be subtle in high frequency input, but it is much more apparent when applied to textures with strong features or sharp edges.

Efros and Freeman [14] proposed a different method for obtaining irregularly shaped patches.

They transform the overlap region between a new patch and the already synthesized texture into an error map, and then use Dijkstra's algorithm [10] to find a lowest cost path through the error map from one boundary to another. This is called the minimum error boundary cut, and it attempts to shape the patches so as to minimize the error along the boundary path. In so doing, the minimum error boundary cut helps to preserve localized high-frequency structure such as edges within the input texture. This makes it a useful technique for approaching textures with distinct, localized features.

Other methods have been used to shape irregular patches, such as graph cuts [28]. This approach uses an energy minimization function to create the patches. These patches are then pasted into the output so as to minimize the appearance of seams, and can even be placed in regions that have already been synthesized.

Overlap repair methods have also been proposed that use a combination of the patch and pixel-based approaches [35, 36]. Nealen and Alexa proposed a hybrid method that uses patch sampling for initial synthesis, followed by a post-process stage where individual pixels are re-synthesized in a carefully calculated sequence. These sorts of methods can yield very compelling results, and can be applied to any of the patch shaping methods described earlier. For the sake of completeness, it is important to note that pixel re-synthesis could be adapted and applied to results generated using a minimum error boundary cut derived from the non-scalar distance metric we propose in order to achieve comparable results.

We have demonstrated that the non-scalar distance metric can easily be implemented within the path planning step used in image quilting in order to achieve a superior minimum error boundary cut. This cut reduces the sharpest discontinuities by encouraging the resulting path to wind around them. In addition to texture synthesis, an improved minimum error boundary cut shows promise in several other domains, such as Wang Tiles and lapped textures, to name only a few.

Wang Tiles are defined as a set of squares in which each edge of each tile is colored, and these colored edges are matched and aligned to tile the plane [53, 54, 8]. Textures can be mapped to Wang Tiles in order to perform a form of synthesis. Cohen et al. have presented a novel means of generating Wang Tiles based on deploying the minimum error boundary cut on an input texture [8].

Another area of interest is lapped textures [41], where texture patches are placed repeatedly onto an arbitrarily shaped surface. While some methods use alpha blending to mix the patch with the surface [41], we believe that using the minimum error boundary cut could potentially be useful for the class of textures to which image quilting is well suited, such as those with strong and distinct edges.

Chapter 3

Modeling Dendritic Structures

3.1 Survey of Existing Techniques

For the sake of comparison, we will survey some of the more common existing procedural techniques for modeling dendritic structures. We evaluate these methods on three important criteria for modeling dendritic structures to achieve artistic effects. After doing so, we will introduce our own method, which performs better than any of the previous methods with regards to the criteria that will now be presented.

Modeling dendritic structures for artistic effects requires first and foremost precise control over the evolving structure. There need to be mechanisms in place to guide where exactly the dendrites should be allowed to grow so that the intended artistic effect is not compromised. In the example of an illuminated manuscript, the decorative dendritic vines should not be allowed to grow across the page such that they obscure any of the text. In addition to this local control, we need to have the ability to govern the global dimensions that the dendritic structure attains. Once again using the example of the illuminated manuscript, we want to be able to ensure that the decorative dendrites cover the border of a page. In order for a procedural method to offer good control for generating artistic effects, it must have both global and local control handles, and we must be able to predict the sorts of results that will emerge when changes are made to these control handles. We want to find control handles that are responsive enough such that the algorithm will be able to create satisfying output using only minimal input, such as characteristics inferred from an input image.

The second criterion is the aesthetics of the approach, and whether the results generated through it resemble compelling branching structures. We acknowledge that different dendritic structures tend to display different characteristics, as can be seen when comparing crack patterns with tree branches, for example. With this in mind, we are looking for aesthetics that are flexible enough to encompass a wide range of dendritic structures.

Finally, we are also interested in the efficiency of a given method. We prefer methods that use minimal computational resources, and can generate their results very quickly. It should be noted that this final criterion is much less critical than the preceding pair.

The techniques listed below will be evaluated on these three criteria. Where possible, this will

be conducted by using them as an image filter to transform an input image into a line art form built out of dendrites. Line art is an artistic medium that includes only distinct straight and curved lines without graduations in shade or color. It is primarily intended to emphasize form and outline. These exacting control requirements make it a good context in which to test the artistic potential of these methods.

The four techniques for generating dendritic structures that we survey all share a common framework that requires the introduction of some basic terminology. They require that space be discretized into a grid that we refer to as the lattice. In our case, this will be a two dimensional uniform grid, although this is by no means a necessary constraint. This lattice is comprised of four- or eight-connected nodes. We refer to the connections between these nodes as edges. In the images below, each pixel represents a node. Nodes are colored black when they are considered to be part of the growing structure. Any nodes adjacent to the growing structure that have not yet been added to it are considered to be part of the frontier. Though each of the techniques surveyed below uses this basic framework, they differ in the way that the structure expands into the frontier.

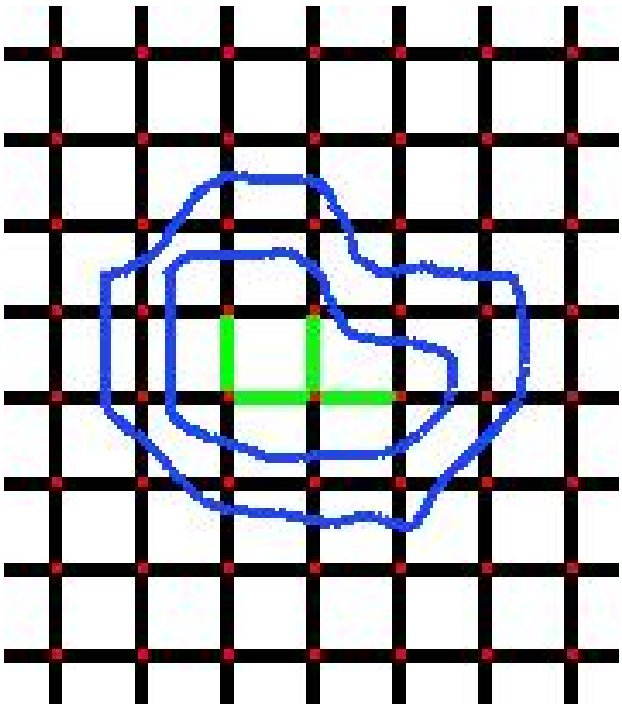

Figure 3.1: A lattice of four-connected nodes

Figure 3.1 shows an example of a lattice containing four-connected nodes. The nodes in green are considered to be a part of the growing dendritic structure, while the nodes that have been circled are adjacent to it, and are considered to be part of the frontier. It should be noted that this lattice is shown at a much lower resolution than the ones we use to grow our dendritic structures,

where each node is approximately the size of one pixel. We typically use lattices of 500x500 nodes to generate our results.

3.1.1 The Eden Model

The basic Eden model is no longer a commonly used method, but it is included here for the sake of completeness. It is one of the earliest procedural dendritic generators [6, 3], and is extremely simple. Using the Eden model, the structure expands to a new randomly selected node from the frontier during each iteration. This leads to a structure that grows almost uniformly, as seen in figure 3.2.

Figure 3.2: The Eden Model

Because there is no complicated calculation for the expansion into the frontier, the Eden model is actually quite efficient. However, it offers no real means of controlling the growth of the dendritic structure. Furthermore, the results do not contain the sort of compelling branches that characterize dendritic structures. Instead, the structure expands in a near-uniform fashion in all directions.

Given the lack of control handles within the Eden Model, no attempt was made to use this method as a line art filter. The lack of control makes this method completely unsuitable for our purposes.

3.1.2 The Modified Eden Model

The Modified Eden Model is a variant of the aforementioned Eden Model that offers more power than its predecessor, at the expense of a minor increase in complexity. The Modified Eden model also expands randomly into the frontier, but it allows us to specify a probability distribution over the lattice, which offers some means of controlling the growth of the structure [3]. Because of this probability distribution, the nodes in the frontier might not be equally eligible to become part of the structure. In figure 3.3, a probability distribution has been specified that encourages the

structure to grow towards the top of the image.

Figure 3.3: The Modified Eden Model

Unlike its primitive predecessor, the Modified Eden model does offer at least some control over the growth of the dendritic structure. However, specifying a probability distribution is not a very powerful control handle. It is difficult to predict exactly what sorts of results will emerge when small changes are made to the probability distributions. Furthermore, this modicum of control is strictly of the local variety. The Modified Eden model offers no way of selecting the endpoints for the dendritic branches. This lack of global control means that although we can influence where the structure will grow during any specific time step, we have no way of controlling the branch's ultimate destination.

The aesthetics of this method are not very compelling. The method of expansion does allow it to vary from the near-uniform shape taken by the Eden Model in accordance to its probability distribution. However, specifying a distribution that would lead to compelling dendritic branches would be a challenge in and of itself.

While this method suffers on control and aesthetics, it does offer competitive efficiency. It has to perform a probability calculation in order to expand, making it less efficient than the original Eden Model, but it can still yield rapid results.

Once again, we found that this method was unsuccessful in serving as a line art filter. We could establish some limited form of control by setting the probability distribution using the gradient edges within the input image, but the lack of global control made it difficult to preserve the important features of the input image. This deficiency makes the Modified Eden model unsuitable for creating artistic effects.

3.1.3 Invasion Percolation

Invasion percolation is a method inspired by fluid flow [50]. It assumes that all the edges within the lattice have been given a weight, and it expands into the frontier along the edge with the lowest weight at each iteration. Figure 3.4 shows the results of invasion percolation with a uniform random distribution of edge weights.

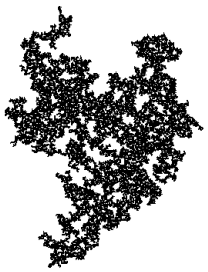

Figure 3.4: Invasion Percolation

This technique offers extremely good local control over the growth of the branches. By specifying appropriate edge weights, we can ensure that the structure grows along a certain path. Furthermore, invasion percolation can be executed efficiently by storing the frontier nodes in a heap data structure. This allows us to grab the node which can be reached with the lowest edge cost in a single operation.

However, using invasion percolation brings forward two major disadvantages. The first is that there is no handle for global control of the growing structure. The weights allow us to determine how the branches expand in a local region, but the structure has no notion of the lattice aside from what is currently on its frontier. This means that the structure cannot easily be directed to reach any specific location on the graph. This is the same weakness that afflicts the Modified Eden model. Figure 3.5 demonstrates how a lack of global control makes it much more difficult to capture all the features in an input image. We can see in this figure that the invasion algorithm might miss entire regions of the image, and there is no mechanism within the method to ensure that this will not occur.

The second limitation is that the branches created by the invasion percolation process are not very compelling from an aesthetics viewpoint. They do not tend to feature the distinct secondary branches that one would expect to see in the natural phenomena that inspires these structures. Instead, the structure tends to grow in clumps that more closely resemble vegetal matter like lichen. This might not always be detrimental, but the fact remains that the invasion percolation framework provides little means of altering this property.

Figure 3.5: The limitations of Invasion Percolation shown by using an edge map (left) to set the edge weights used in invasion percolation (right).

The first panel of figure 3.6 shows the result of using invasion percolation as a line art filter on the Coca-Cola logo. The edge weights in the lattice were set to be lower where gradient edges exist in the corresponding part of the input image. We can see that in this specific case, the lack of global control does not cause invasion percolation to miss many of the important features in the image. However, the clumpy aspects of the resulting structure do not strongly resemble the dendritic branches that can be achieved using other methods.

3.1.4 Diffusion-Limited Aggregation

Diffusion-limited aggregation (DLA) is one of the most common procedural technique for modeling dendritic structures [5, 6]. This algorithm is meant to simulate the aggregation of dust particles, and proceeds by releasing 'random walkers' from a long distance away from the structure. These walkers move randomly through the lattice until they are adjacent to the structure, at which point they will attach themselves and a new random walker will be released. As seen in figure 3.7, this yields extremely compelling results that very much resemble branches we would expect to see in the real world. Pseudo code for DLA is shown in figure 3.8.

The primary control handle in DLA involves the positions of the generators that launch the random walkers. The structure will tend to grow towards these generators, and this property allows us to achieve a level of global control absent from any of the preceding methods. Unfortunately, basic DLA offers no real form of local control. We can direct the eventual outer boundaries of the structure by selecting the generators, but we have no way to control what path is used to get there, unless we use a more complex form of DLA that includes stickiness. The results of this effect can

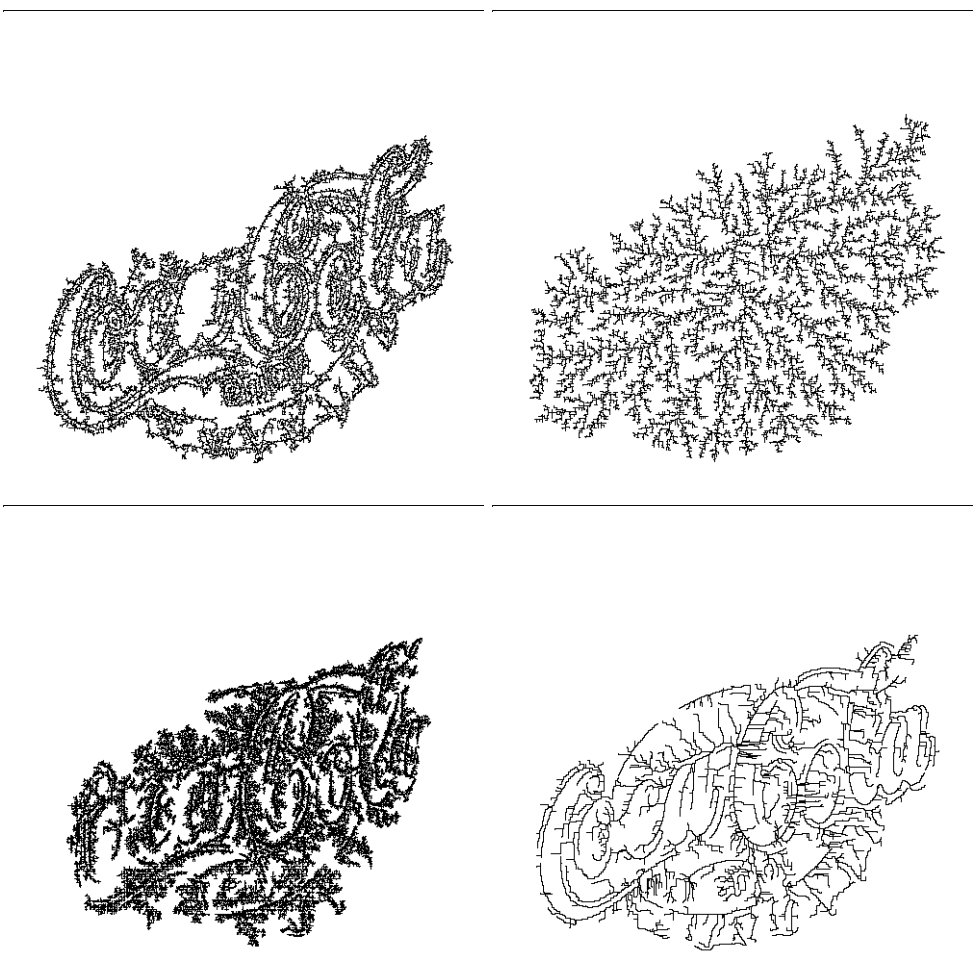

Figure 3.6: Dendritic patterns resembling the Coke logo created with invasion percolation (top left), diffusion-limited aggregation (top right), diffusion-limited aggregation with stickiness (bottom left) and our path planning method (bottom right).

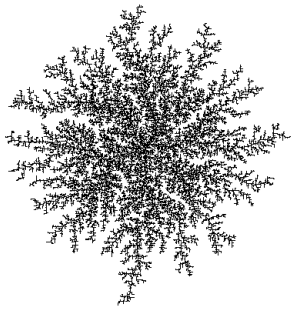

Figure 3.7: Diffusion-Limited Aggregation

be seen in the second panel of figure 3.6.

Efficiency is also a major concern when executing DLA. Releasing particles and allowing them to wander freely throughout the lattice until they find the structure comes at a considerable speed cost, especially in the early stages of the algorithm. Several tricks and optimizations can be used, such as discarding points if they leave a certain radius around the structure, but it remains a slow process.

When applying DLA as a line art filter, we were able to leverage its global control characteristics by selecting our generators as a proportion of the gradient edge pixels within the input image. This allows us to effectively grow the structure out towards these generators, as shown in the second panel of figure 3.6. However, the lack of local control in basic DLA leads to the destruction of all the interior detail within the structure, a property that makes using it for artistic effects all but impossible.

The DLA concept does have one aspect that allows us to partially compensate for this deficiency. More complex forms of DLA can contain a stickiness factor that determines whether a random walker will actually attach to the structure. In general, a lower stickiness will serve to thicken the branches of the structure, because the particle will sometimes continue to move adjacent to the structure for several steps before actually attaching to it. We are able to address the lack of local control by basing the stickiness on the gradient edge pixels within the input image. This means that a random walker is much more likely to stick on top of an edge in the input image, providing us with some minimal control over the paths taken by the DLA branches, as seen in the third panel of figure 3.6. However, it should be noted that stickiness only exacerbates the efficiency issues that afflict DLA. Furthermore, the compelling nature of the results generated by the DLA method is compromised when using stickiness. Figure 3.8 shows pseudo code for the DLA algorithm including stickiness.

Figure 3.9 shows that the global control present in DLA allows us to better capture features

```
DLA()
{
  // Create a particle at a generator and release it into the graph.
  // We can continue repeating this step until we have released an
  // arbitrary number of particles.

  Particle p = generateParticle();

  while (p is not attached to the structure)
  {
    Move p along a randomly selected adjacent edge.


    // We next check if p is adjacent to the structure, and thereby
    // in the frontier.

    if (any of p's neighbours == occupied)
    {

      // If we are using stickiness, we perform a stickiness test to
      // determine if the particle has attached to the structure.

      if (random_value < the stickiness factor at this location)
      {
         attach p to the structure
      }

    }
  }
}
```

Figure 3.8: Pseudo code for the DLA algorithm including a stickiness factor.

within the input image. By comparing this with the results generated from the same input using invasion percolation (shown in figure 3.5), we can see that DLA's global control gives us a means of ensuring that we capture all the important features in this image. The example shown in this figure also uses stickiness to achieve some modicum of local control.

3.1.5 Discussion of Previous Methods

Control is the most important aspect for generating artistic effects. Although some of the four procedural methods – particularly diffusion-limited aggregation (figure 3.7) – were able to produce very compelling results, they remain extremely difficult to control. In the case of DLA, we find that the compelling aesthetics – the main advantage of this method – have to be compromised in order to achieve any sort of local control. Figure 3.6 shows the results of the more advanced techniques when applied as line art filters.

Although the existing procedural methods might suffice for certain applications, their lack of

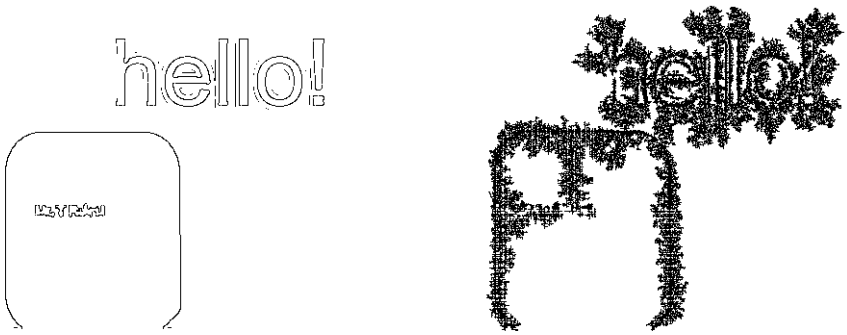

Figure 3.9: The benefits of DLA's global control shown by using an edge map (left) to serve as generators for DLA with stickiness (right).

intuitive control handles makes them impractical for generating artistic effects. In the next section of this thesis, we introduce our own procedural approach for modeling dendritic structures using path planning. This path planning approach offers excellent local and global control handles and can be computed in a mere fraction of the time taken by DLA, as shown in figure 3.10. The method we propose is still in its infancy, so it is likely that the aesthetics it provides will improve substantially as it becomes more mature, but we believe the current state is comparable to some dendritic structures that exist in the real world. Output from a line art filter created using our path planning method is shown in the bottom right of figure 3.6.

Figure 3.10 shows a timing analysis of the methods that were described above. We have measured the time required for each approach to model a dendritic structure that represents a line art version of the Coca-Cola logo. The times in seconds are displayed when modeling structures of several sizes (where the size is determined by the number of lattice points that are part of the dendritic structure) to show how the different methods scale. The numbers that are underlined show the times needed to create the reasonably faithful representations of the Coca-Cola logo shown in figure 3.6. Several of the methods were unable to create a reasonable approximation of the image using a small number of particles, and as a result those timing figures have been left blank.

Invasion percolation is by far the fastest of the methods shown here. The expansion criteria is very simple, and can be accomplished with one operation when a heap data structure is used to store the frontier. On the opposite end of the scale, DLA and its version with stickiness are both extremely time consuming. Our path planning approach offers fast speeds that scale well as the number of particles increase. Furthermore, path planning can create a reasonable approximation of the source image with far fewer particles than any of the other methods. The other methods end

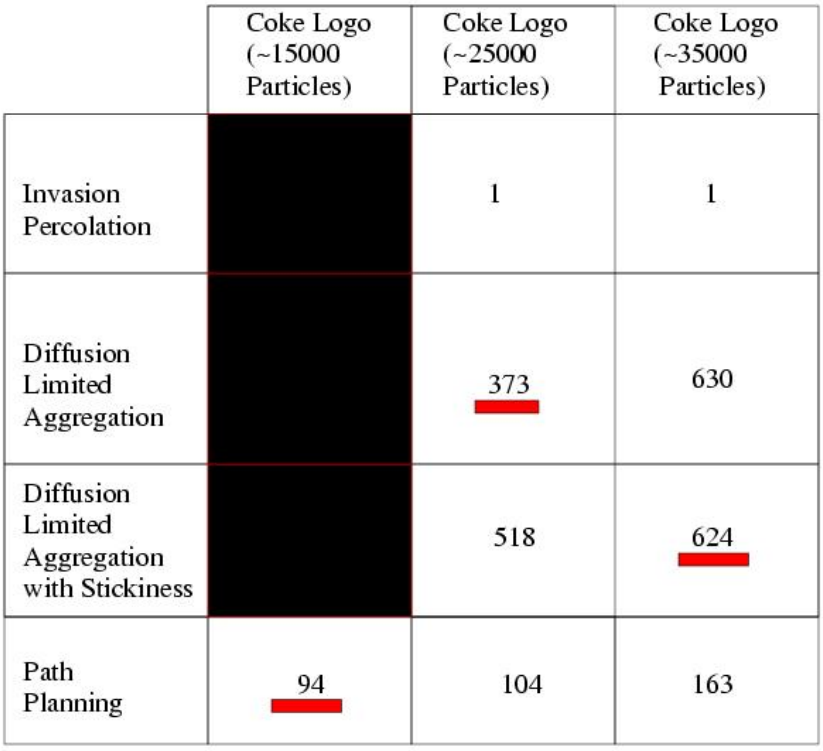

Figure 3.10: Timing analysis measured in seconds.

up 'wasting' particles by thickening branches and covering unnecessary space, resulting in higher total times.

3.1.6 Modeling Dendritic Structures with Path Planning

Now that we have surveyed some of the more prevalent existing procedural techniques for modeling dendritic structures, we present our own solution to this problem. Our method involves the use of path planning, and we believe it captures the best characteristics from the preceding methods. The first part of this section will describe our path planning algorithm for dendritic growth, while the second will introduce the partial non-scalar distance metric and show the advantages it presents for achieving artistic effects.

3.2 Path Planning Algorithm

The path planning algorithm we have applied to generating dendritic structures is extremely simple, which is another one of its advantages. Drawing inspiration from the artificial intelligence technique of the same name [58, 10], it requires users to specify both a set of endpoints and weights for all the edges in the lattice. We can achieve dendritic properties – that no branches cross over each other – by assuming that all paths originate from the same initial seed point. This is demonstrated by the structure shown in figure 3.11.

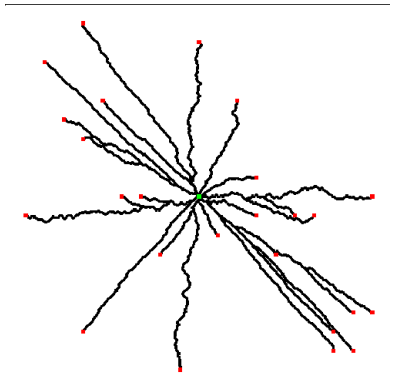

Figure 3.11: Modeling a dendritic structure using path planning

A representation of a lattice of 4-connected nodes is shown in figure 3.1. We use a uniform grid for our path planning approach that contains 8-connected nodes, though it is important to note that our method would work on any arbitrary graph. In the examples shown below, each node is considered to be the size of a pixel. Each node is given an initial weight, and the edges are weighted as the average weight of the two nodes that they connect. In most of our examples, we set the weights of our nodes based on characteristics at that location in an input image. An equation specifying how this can be done for dendritic halftoning is shown later in this thesis. It should be noted that the weights can be chosen in many other ways, such as by drawing them from a random distribution.

The first step in our method is to perform the brushfire algorithm on the weighted lattice. The purpose of the brushfire algorithm is to determine the 'shortest' distance from the initial seed point to every other node in the lattice. The classic path planning algorithm uses the cumulative distance metric to determine the shortest cost to reach a node. This means that the cost to reach a node is the sum of the edge weights that have been traversed to reach that node. At each step, the brushfire algorithm grabs the node from the frontier that can be reached with the lowest cost. All of its neighbors and the edge weights to reach them are then added to the frontier. The algorithm concludes once all the nodes in the lattice have been visited and assigned the 'shortest' cost needed to reach them. Pseudo code for our brushfire algorithm is provided in figure 3.12.

```
brushfire()
{
  // The partial paths are stored in a heap datastructure which allows
  // us to grab the shortest partial path according to the distance
  // metric being used in an O(1) operation.

  // We start from an initial seed point at an arbitrary position in
  // the graph.

  Path initial.current_point.coordinates = any arbitrary seed point coordinates;
  initial.cost = 0;

  heap_push(initial);

  while (heap is not empty)
  {

    // Gives us the shortest prospective path.
    Path path = heap_pop();

    // We investigate the path if the point has not already been
    // visited by the brushfire algorithm.

    // We grab the current_point from the graph using its coordinates.
    Point current_point = graph_points[path.current_point.coordinates];

    if (current_point.visited == FALSE)
    {

      current_point.visited := TRUE;
      current_point.shortest_cost := path.cost;

      for (each edge current_edge connected to current_point)
      {
        // We add a partial path along this edge if the point at the
        // other end of it has not been visited.

        if (current_edge.other_point.visited == FALSE)
        {
          partial_path := path;

          partial_path.current_point.coordinates := current_edge.other_point.coordinates;

          partial_path.cost := path.cost + current_edge.cost;

          // We push partial_path onto the heap now that it has been
          // adjusted to show its current parameters.

          heap_push(partial_path);
        }
      }
    }
}
```

Figure 3.12: Pseudo code for our brushfire algorithm.

This preprocessing brushfire step only needs to be done when the set of edge weights is changed. The addition or variation of endpoints does not require any changes to the costs needed to reach any of the nodes in the lattice. This allows us to generate many different structures with very little overhead.

Once the brushfire algorithm has finished, we can turn to generating the paths. Instead of expanding the structure out into the frontier, the path planning approach traces backwards from each endpoint towards the single seed point of the structure. We can further optimize this method by calculating the paths only until we reach a part of the structure, at which point we already know the shortest cost path back to the seed point since we calculated it when pathing back from a previous endpoint.

Path planning seems to take the best characteristics of all the methods surveyed above, because it allows us to retain the local control of invasion percolation while also capturing the advantages of the global control that are seen in DLA. Since we have the power to set end points for the paths, in addition to the edge weights throughout the lattice, we can control both where the branches go and how they get there, and we can generate them very quickly without any sort of human intervention.

3.2.1 The Non-Scalar Distance Metric

While path planning offers good control handles for artistic effects, we find that the cumulative distance metric that is classically used to determine the shortest path (where the cost of a path is equal to the sum of the costs of all the edges that have been traversed) has some adverse consequences for our applications, as shown in figure 3.13. In particular, we find that the paths generated using this distance metric will tend to take short cuts over high cost areas in order to avoid long, winding routes. Not only is this bad from a control standpoint, as such short cuts can ruin high frequency details that are important for artistic effects, but we also believe that long, winding paths more closely resemble stylized dendritic structures, as seen in figure 1.1.

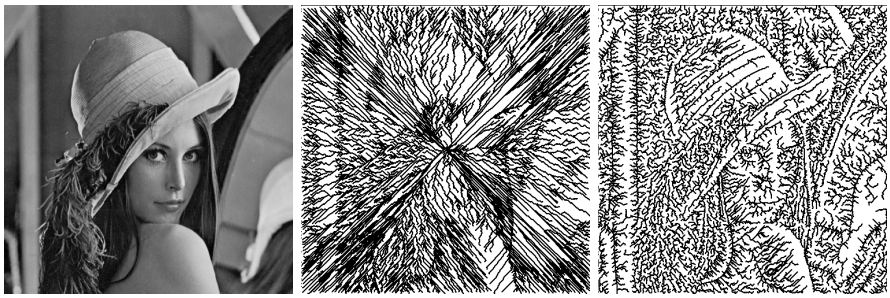

Figure 3.13: Source image (left) and a dendritic version using the cumulative distance metric (center) and the partial non-scalar distance metric (right).

As such, one of our main contributions is the application of a non-scalar distance metric first introduced in the field of robotics [38]. This non-scalar metric can be implemented by storing the list of edges for a prospective path in the order of decreasing cost. When two prospective paths are being compared, the corresponding element from each array of edges is selected. As soon as one differs from the other, the path with the smaller value is chosen as being the 'shorter' of the two. In essence, this metric can be seen as minimizing the maximum edge costs in the path.

Sorting the edges along a path in descending order is not a cheap operation. While there is some speed penalty, the main cost comes in the form of memory. Path planning algorithms tend to use a lot of memory, and this is no exception. Storing a list of edge costs needed to reach each point is a fairly prohibitive cost. Fortunately, we find that we can achieve results similar to the full non-scalar distance metric using only a partial version.

The partial non-scalar distance metric implemented in this thesis keeps track of only the maximum edge cost traversed along the path. Comparisons between paths are first made by comparing these two maximal cost values. In the case where there is a tie, we then rely on cumulative path cost to determine the shorter path.

```
partialNonscalarCompare(Path p1, Path p2): Path
{
  if (p1.maxEdgeCost == p2.maxEdgeCost)
  {
    return the path with the lowest cumulative distance;
  }

  else
  {
    return the path with the lowest maximum edge cost;
  }

}
```

Figure 3.14: Pseudo code for the comparison of two paths using the partial non-scalar distance metric.

Pseudo code for the comparison between paths for the partial non-scalar distance metric is shown above. It returns the 'shorter' path by first comparing the maximum edge values in each, using cumulative distance only to break ties. As shown in figure 3.15, even this very reduced version of the non-scalar distance metric produces markedly different results from the normal cumulative distance metric.

Figure 3.16 shows another way of visualizing the differences between the cumulative distance metric and our partial non-scalar distance metric. This figure includes visualizations of two graphs where the edge weights were drawn from uniform random distributions, using the center as the seed point. These visualizations show the effect of path planning to every node in the graph from the

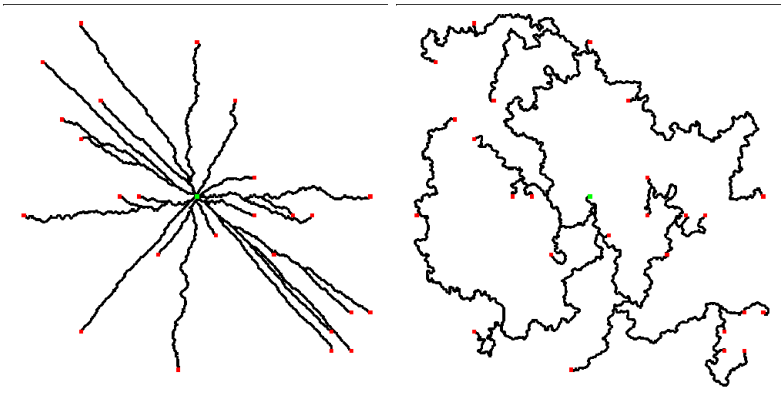

Figure 3.15: The conventional cumulative distance metric (left) and the partial non-scalar metric (right) using the same set of 28 endpoints in the same weighted graph.

seed point, and nodes are colored more brightly each time they fall along a path. This once again demonstrates that the cumulative distance metric is dominated by proximity to the seed point, while the non-scalar distance metric contains more winding and irregular paths.

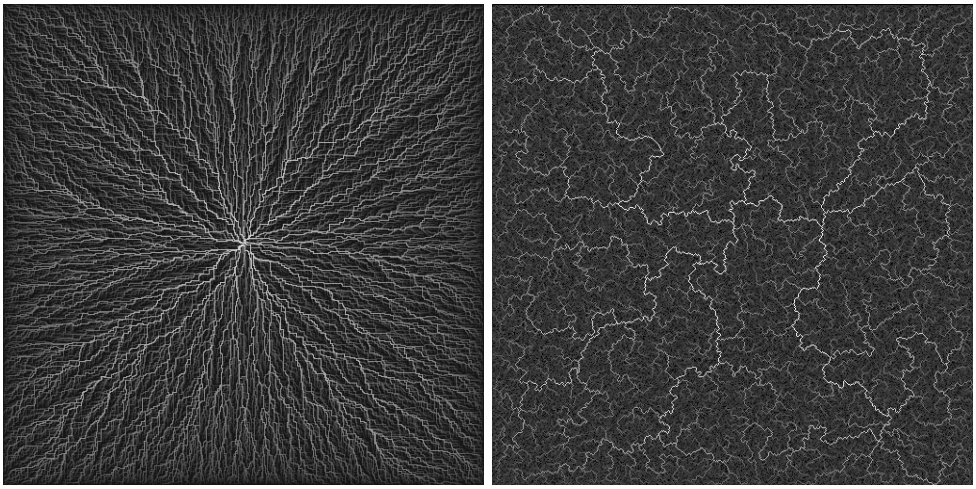

Figure 3.16: Visualizations of a graph using the cumulative distance metric (left) and the partial non-scalar distance metric (right).

We find that using the partial non-scalar distance metric instead of the traditional cumulative distance metric has little impact on the efficiency of the overall algorithm. The dominant step of the pathing operation is the brushfire algorithm that is used to find all the costs in the graph. The partial non-scalar distance metric that we use to compare paths during the brushfire step requires only one extra operation. The paths generated by our method do tend to be longer than the ones created using the cumulative distance metric, but this does not have a large effect on the speed

of the algorithm. As such, we conclude that modeling methods based on our approach will be comparable in speed to path planning using the traditional cumulative distance metric.

In addition to the empirical data shown in figure 3.10, we have also formalized the timining requirements of our algorithm. The brushfire step will visit each of the n nodes in the lattice. After visiting each node, the brushfire algorithm will pop the next node to be visited from the frontier heap, which is a log(m) operation where m is the number of nodes in the frontier heap. The timing analysis of our algorithm is O(n * log(m)). The greedy algorithm that traces the shortest paths back from each endpoint is O(l) where l is the length of the path being traced.

3.2.2 Controlling the Dendritic Growth

Controlling the growth of the dendritic structure is of great importance. As mentioned previously, we believe that path planning offers very responsive control handles. In particular, we wish to be able to control the evolving structure based on an input image. Though it is important to note that our technique can be used on graphs of arbitrary shape and size, we find that artistic effects are most commonly applied as image filters - in our case, an input image is transformed into a dendritic version that simulates a certain artistic style, be it halftoning, line art, ornamentation or a host of others.

Local control can be obtained by setting the edge costs according to some parameters drawn from the same location in the input image. In most cases, a combination of the gradient and intensity values is helpful for preserving the structure of the image, though the exact combination is dependent on the effect being sought. The endpoints will determine the global dimensions of the dendrites, and should be chosen as the most 'important' points in the image – again, the scale of importance is based on the context.

Figure 3.13 shows an input image and dendritic structures created using the conventional cumulative distance metric and the partial non-scalar distance metric proposed in this thesis. Although both feature the same control handles, the partial non-scalar distance metric adheres to them more rigorously. This is because the paths generated by our partial non-scalar metric are not as obsessed with the need to take the fewest number of edges to reach their destination, as is seen with the cumulative distance metric.

The different ways that the two metrics respond to the same control handles are demonstrated once again in figure 3.17. This is another visualization (similar to figure 3.16) in which the weights in the graph have been set using information from an input image. The cumulative distance metric's dependence on proximity to the seed point destroys many key features in the image. In contrast, the partial non-scalar distance metric is able to preserve the main structure of the input image.

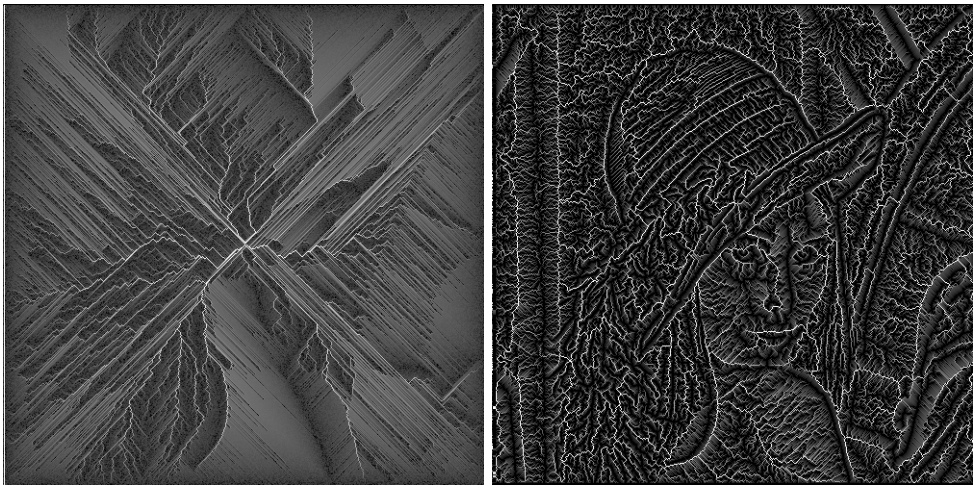

Figure 3.17: Visualizations of graphs built from the input image of Lena (from figure 3.13) using the cumulative distance metric (left) and the partial non-scalar distance metric (right).

3.2.3 Dendrified Geometry

For most of this thesis, we apply our path planning approach to modeling dendritic structures on a uniform two-dimensional lattice. However, it is important to note that this method is not limited to these parameters. In fact, our method will succeed in growing dendritic branches across any arbitrary graph of connected nodes.

This means that we are not limited to two-dimensional applications. We can grow dendrites across arbitrary geometry by defining the vertices in the geometric description as the nodes in our lattice. The edges between the vertices can provide the connectivity for our lattice.

Figure 3.18 shows three-dimensional models that have green dendritic structures spreading across their surfaces. This approach will work with any arbitrary model, although the distance between vertices limits the amount of high frequency detail along the branches. This problem could be circumvented by re-sampling the model's mesh at a finer level.

Our work in this area is only preliminary, but we feel there are several possible applications. One application implied by our example would be the growth of lichen or other vegetation across objects. We also believe this approach has potential for modeling decorative ceramics, which are often covered with rich dendritic ornamentation.

3.3 Dendritic Halftoning

In this section, we focus on the use of dendrites for achieving artistic halftoning effects. Our stylized dendrites are not intended to match the tone of an image, but instead to create compelling results

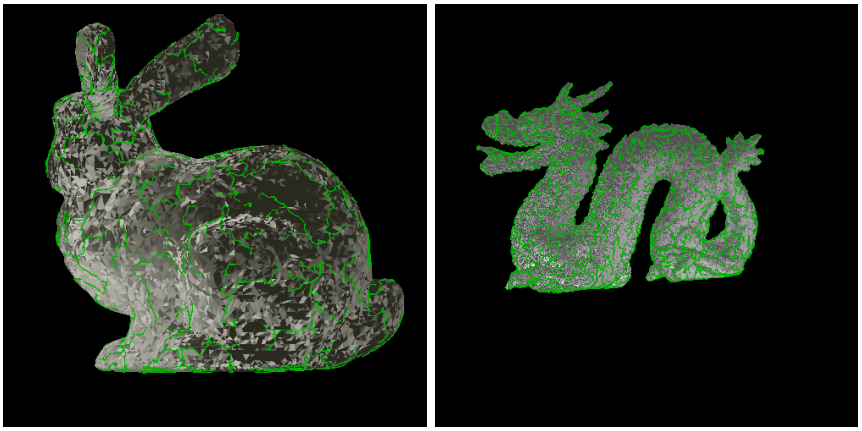

Figure 3.18: Three-dimensional models with dendritic structures grown across their surfaces.

by emphasizing gradient edges and contrasts that are present in the input image.

Many traditional media use halftoning processes to enhance the aesthetics of their results, such as pen and ink drawing, shading, and stippling, to name only a few. These artistic effects employ different halftoning primitives in order to achieve their aims. We propose the idea of dendritic halftoning, which treats the branches of a dendritic structure as halftoning primitives. The resulting structure can then be rendered using any sort of dendritic metaphor, ranging from cracks to vegetal growth.

Existing methods for stylistic halftoning tend to focus on tone analysis and matching, and are not well suited for representing sharp features in the input image. Our method is designed with the opposite ideology, wherein we recognize that the human visual system perceives contrast in edges more than precise tone values. Guptill's book on pen and ink illustration [21] supports this notion and suggests that artists tend to create their desired effects by emphasizing edges rather than solely by seeking to match specific tones.

Previous procedural methods for pen and ink rendering have not adequately addressed this aspect of halftoning. Some approaches have tried to recognize gradient edges by clipping strokes to region boundaries [47], but this leaves only an implied edge, visible because of an omission of strokes. We note that artists would actually tend to draw strokes along these boundaries, instead of just terminating their strokes in that vicinity. Our algorithm provides a mechanism for accomplishing this, using dendrites as our primitives.

Having described our path planning algorithm in a previous section, we are now ready to discuss how it can be applied to the problem of artistic halftoning. There are several steps in this process, including the selection of edge weights, the placement of endpoints, and the application of different rendering styles that can be used to achieve different effects and emphasize important edges in the

input image. Each of these will be described in detail below.

3.3.1 The Importance Map

Several procedural halftoning approaches use the notion of an importance map to distribute strokes or primitives throughout the input image [51]. We build each pixel of our importance map as a combination of the intensity and the gradient magnitude at the corresponding pixel in the input image. Figure 3.19 shows how we combine the intensities in the input image along with the gradient magnitude of that image to create our importance map. Equation 3.1 shows that the importance of a pixel at (i, j) is a function of the gradient and intensity at that point. GW and IW are constants that represent the weight we place on gradient and intensity, respectively. We have found that a GW of 35 and an IW of 10 tend to work well.

$$Importance(i,j) = (GW * Gradient(i,j))/(GW+IW) + (IW * Intensity(i,j))/(GW+IW); \quad (3.1)$$

Halftoning is intended to preserve color contrasts, even while reducing the color resolution of the image. As such, we ascribe high importance to dark pixels in the input image, and to areas that fall along strong gradient edges. The importance map shown on the right in figure 3.19 represents areas that are more important with lighter intensities.

Using the importance map to control the evolution of a dendritic structure is rather simple. We can coerce the branches to follow important gradient edges and to enter dark areas by assigning the edge weights in the graph based on the importance of adjoining nodes and a small random factor. More important nodes will feature lower edge costs, making them more likely to be traversed by the growing structure.

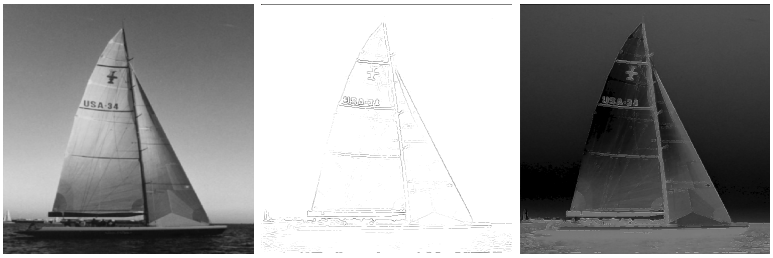

Figure 3.19: Source image (left), gradient magnitude of the source image (center), and the resulting importance map. (right).

The weights in the graph will determine the paths along which the dendrites will travel to reach the endpoints. Part of our goal is to emphasize strong gradient edges in the image, a prospect that is difficult to achieve using traditional halftoning techniques. We can ensure that the dendrites

will follow the gradient edges by making the edge weights in the graph mostly dependent on the gradient magnitude, and by lowering the random contribution.

Each node in the graph is given a weight. For most of the halftoning examples shown in this thesis, this weight includes a contribution from both the gradient and the intensity. We also employ a small random contribution to the weight of each node. The images were best captured when this random contribution was only 2 or 3. This seemed to be sufficient to provide variation in the image without seriously damaging the appearance of the input image. Equation 3.2 shows how the weight at each node is calculated. K represents a constant value that is added to the weight, while GW and IW represent the weight that is placed on the gradient and the intensity. We have found that K=20, GW=35, IW=10 and R=2 lead to good results.

$$Weight(i,j) = K + GW * Gradient(i,j) + IW * Intensity(i,j) + random()\%R; \tag{3.2}$$

We already have enough information to determine that our non-scalar metric responds more appropriately to the control handles inherent in the path planning approach. The formulations above were used in our brushfire algorithm to produce the visualization shown in figure 3.20. The lighter areas represent nodes that are visited earlier in the brushfire, and can be reached at a lower cost. It is easy to see that the cumulative distance metric is ruled by proximity to the seed point, at the expense of the image structure. On the other hand, the non-scalar distance metric is willing to take long, winding paths to reach its destination without crossing high cost areas. This allows it to better preserve the features in the input image.

This is further demonstrated when we build our dendrite. Figure 3.13 shows an input image and dendritic structures created using the conventional cumulative distance metric, a form of which was employed to find features by Mortensen and Barrett [32], and the partial non-scalar distance metric proposed in this thesis. Although both feature the same control handles, the partial non-scalar distance metric does not penalize long paths, which means they are better able to follow features in the image. As a result, the structure of the image is better captured by our approach.

3.3.2 Endpoint Placement

Global control over the evolving dendritic structure can be established by carefully selecting endpoints from the input image. In general, we want to select the most important points in the image to serve as our endpoints. However, employing only such a naive approach would not guarantee any spacing between endpoints, which can damage the aesthetics of the results.

We use Mould's stippling method for selecting our endpoints [34]. This involves a second iteration of the brushfire algorithm that we already used to determine the cost of reaching all the nodes in the lattice. This second iteration of the brushfire algorithm uses the cumulative distance

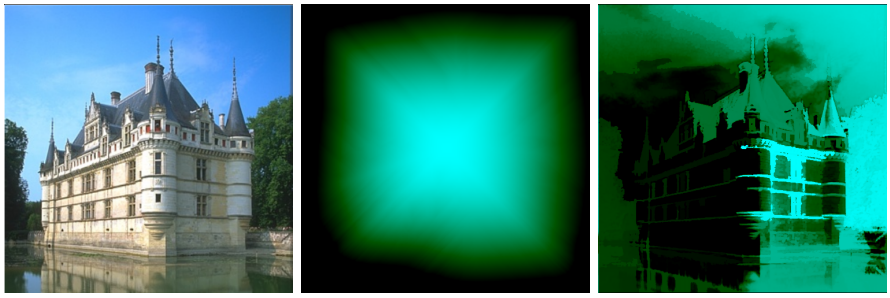

Figure 3.20: Source image (left) and brushfire contours using the cumulative distance metric (center) and the non-scalar distance metric (right).

metric, unlike the previous iteration of the brushfire algorithm, to decide when a new endpoint should be chosen. The endpoint selection brushfire uses the cumulative distance metric because it heavily considers proximity to the seed points when calculating costs, unlike the partial non-scalar distance metric, and this is useful for ensuring a distribution of endpoints with the right amount of spacing that resembles the image.

This algorithm functions similarly to the brushfire shown in figure 3.12. When the total cost to reach the next node in the frontier exceeds a certain threshold, we choose the most important point on the frontier, as per the values in our importance map, as a new endpoint. This node is then considered to be another seed point for the endpoint selection brushfire. The algorithm considers any seed points to be reached with a cost of zero, which means that the brushfire will find it cheap to expand paths from new seeds. The algorithm continues until every node in the lattice has been visited. Most of the dendritic halftoning shown in this thesis uses a threshold of around 1200.

The result of this approach is a set of endpoints that can be used for stippling [48, 34], assuming the chosen threshold is low enough. An example of the endpoints chosen is shown in figure 3.21. As can be seen, the endpoints selected in this manner do a good job of capturing the essence of the input image.

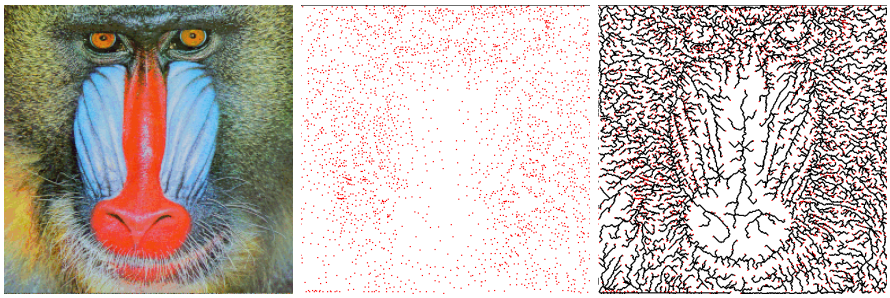

Figure 3.21: Source image (left), the set of endpoints (center) and the resulting dendritic structure (right).

3.3.3 Halftoning Emphasis

To enhance our artistic halftoning effect, we also wish to emphasize certain parts of the image based on their gradient and intensity values. Most procedural artistic halftoning approaches apply some small changes to their primitives in order to accomplish this, ranging from angling strokes to increasing the size of stipples. Our halftoning primitive is a dendritic path, which naturally lends itself to changes in width based on the importance of a region. In this way, we can have thicker branches passing along more critical sections in the input image.

We set up a thickness for each node in the structure using a rolling average of importance along each branch. By taking into account the values of several neighbors on both sides, we are able to achieve a smoother transition between the thickness levels. By allowing this window to overshoot the end of the branches, we are able to cause natural thinning of the dendrites as they near their tips. This property is not desirable for all rendering metaphors, but works well for the vegetal analogy we explore later in this section. Equation 3.3 shows the thickness calculation for a single node. We have found that Z values around 0.1 tend to work well. The final thickness is calculated by averaging the value obtained from equation 3.3 with the thickness calculated at adjacent nodes on the branch.

$$Thickness(i,j) = Importance(i,j)/Z; \tag{3.3}$$

Figure 3.22: Source image (left) and a dendritic version (center) and a dendritic version with halftoning emphasis (right).

Figure 3.22 shows the results of adding halftoning emphasis to our dendritic structure. The varying widths of the branches provide visual cues for the important areas in the input image. These thickness values can be stored along with information about the dendritic structure for further rendering purposes.

3.3.4 Vine Rendering

The sections above describe a general framework for creating a dendritic structure that halftones an input image. The resulting structure can then be rendered as any sort of dendritic primitive. In this thesis, we primarily render our halftoning structures using a stylized vegetal metaphor, but we believe small changes in the modeling process would make the dendritic structures appropriate for many other rendering styles, ranging from lightning to cracks, frost, or any sort of branching primitive. We will later discuss how we can model our structures as lightning, and a version of our dendrites rendered as cracks is displayed in figure 3.23.

Figure 3.23: Source image (left) and a dendritic version rendered as cracks (right).

The first step involved in rendering our structure as stylized vines is a smoothing procedure. We note that stylized vegetal features are generally more smooth than our basic dendritic structure. In order to be able to manipulate the smoothness value of the structure, we wish to set up our dendrites as something similar to a reduced version of multiresolution curves [16]. We accomplish this by representing our structure as a set of cubic b-splines. We can reduce our dendritic representation by using only every nth control point, and allowing the splines to interpolate between them. By adjusting n, we can achieve an arbitrary level of smoothness, as shown in figure 3.24.

Smoothing the dendritic structure results in a more stylized appearance. However, excessive smoothing destroys details in the input image, and can lead to an abundance of straight branch stubs that are not conducive to our aim. We also find that fewer endpoints are necessary for a reasonable stylized representation than are used for our generic halftoning dendrites.

The next step is to render the branches as stylized vines. Using the width information collected while generating the dendrite, it is a simple matter to render the branches in vegetal form. A black outline was added around the dendrites to give them a more artistic look. The results of this process are shown in figure 3.25. In cases where the input image does not dictate many changes in width,

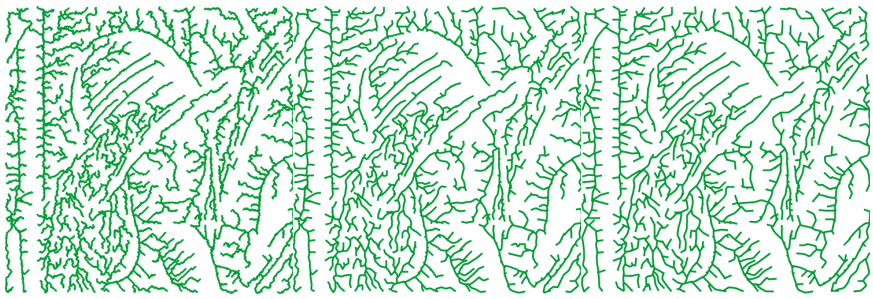

Figure 3.24: Smoothed spline representation of a dendritic structure with n = 1 (left), n = 8 (center) and n = 12 (right).

we can increase the artistic aspects of our results by adding stylistic width. Smooth width changes can be achieved by employing a sine function and setting all negative results to zero, which will cause the branches to gradually oscillate between thin and thick, in accordance with the frequency that we specify.

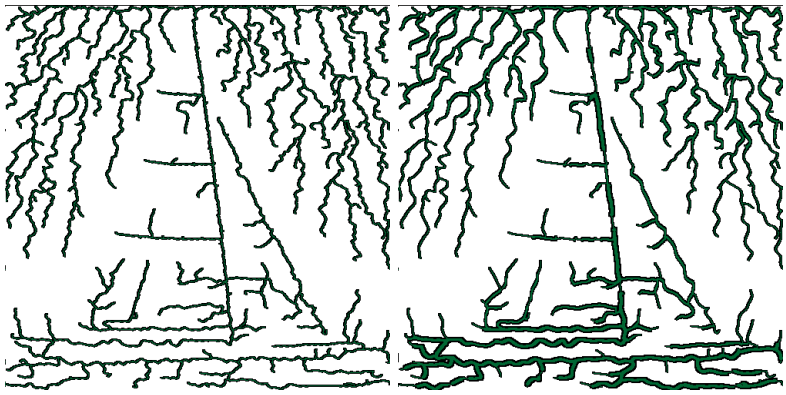

Figure 3.25: Vine rendering of halftoning dendrites without thickness (left) and with thickness (right).

The interaction between decorative dendrites and other nearby features, including other dendrites, is another important aspect to their aesthetic appeal. As shown in figure 3.26, we have done some preliminary investigation into this sort of intertwining behaviour. Our approach is to set up a newly generated dendrite such that it will interact with a background image, be it of a feature such as a letter or a set of previously generated dendrites. The newly generated dendrite is then set to alternate between crossing over and the under the background image at each new intersection with the colored pixels in the background image. This is a very primitive heuristic, but it serves to demonstrate the effects of intertwining dendrites.

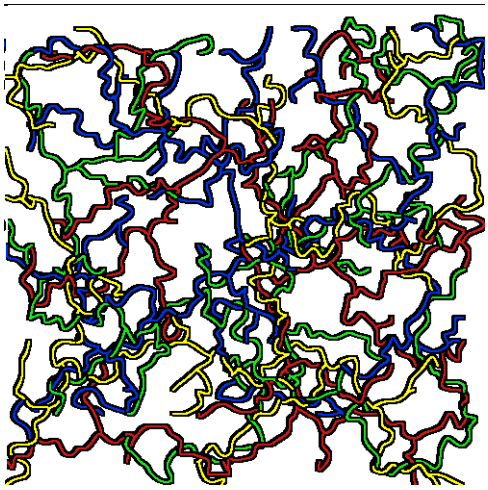

Figure 3.26: Four intertwining dendrites.

3.3.5 Dendritic Halftoning Results

Figures 3.27, 3.28, and 3.30 show some results from our artistic halftoning process. The input images are shown on the left, while the center shows the generic halftoning dendrites. The images on the right show the halftoning dendrites rendered and smoothed as vine structures. The sample of images we include here demonstrate that our method is able to effectively capture even subtle features, such as the weak gradient edges in the cat image in figure 3.27.

Similarly, we note that our dendrites will emphasize features that are barely discernible in the input image. Some of these fine details include the cloth folds in the second row of images in figure 3.28, or the fabric of the sail at the top of figure 3.30. This is because the partial non-scalar distance metric can coerce the paths to avoid locally expensive edges, forcing them to take longer routes along even very weak gradient edges. This allows us to capture small details, assuming there are not more important features in the area to attract the paths. This allows our method to capture subtle details that could be easily missed by other approaches.

Based on the results generated for this thesis, we can make several observations insofar as how well different types of images will weather the dendrification process. In general, our process works best when there is high contrast between different areas of the image. It also succeeds in capturing high frequency detail, as seen with the fur in the mandrill image in figure 3.27 and the feathers on Lena's hat in figure 3.30. It does not perform quite as well in clearly representing smooth intensity gradients, or at conveying absolute intensity, as demonstrated in the sky above the sailboat in figure 3.30. We do not consider this to be a significant drawback, as strict tone matching was not one of our priorities.

Our method seems particularly effective for generating dendritic portraiture, as seen in most of

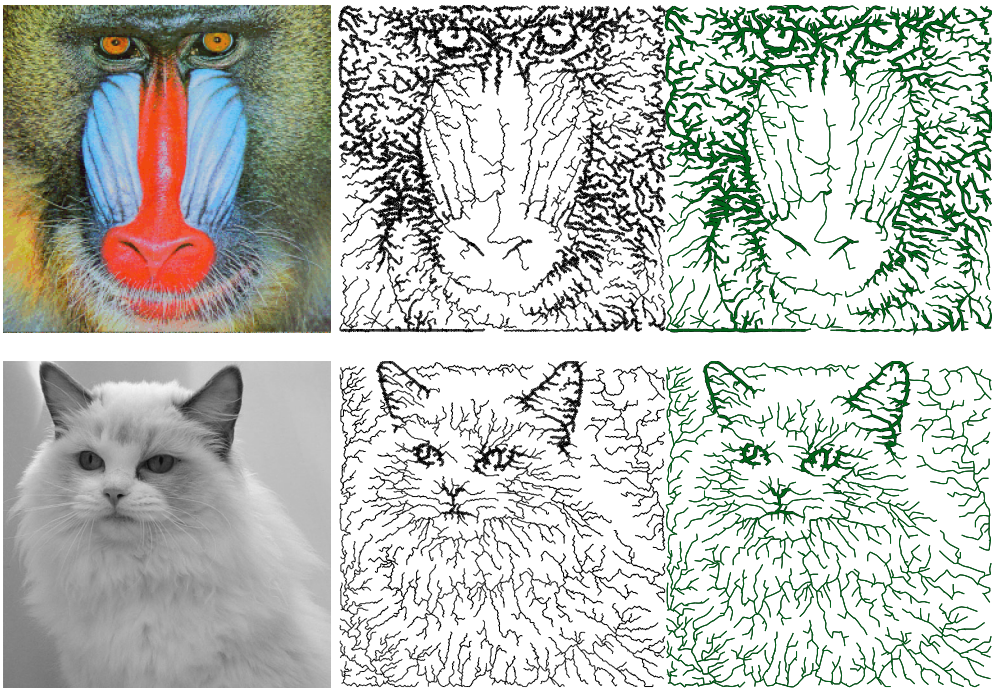

Figure 3.27: Dendritic Halftoning

figure 3.28. Gooch et al. noted that facial recognition and learning can be improved by transforming original photographs into non-photorealistic equivalents [20]. While we have not undertaken a rigorous study in this regard, anecdotal evidence suggests that the results from the dendritic halftoning of human portraits tend to remain quite recognizable.

We have also observed a trade-off that occurs between the effectiveness of the rendering metaphor and the structure of the original image. Trying to achieve the more stylized look that would be characteristic of ornamental vegetation jeopardizes some of the fine detail originally preserved by our halftoning dendrites. Our method provides a mechanism for exploring this trade-off. By adjusting the smoothing variable, a user can choose which aspect they consider more important.

Figure 3.29 shows the time our method took to generate some of the results presented in this paper. The number of seconds required to model the halftoning dendrite is largely dependent on the threshold used for selecting endpoints. For the sake of brevity, we show only a subset of our timing results, as all the images tended to fall within the time range of those listed below.

The previous section demonstrated how we can render our dendritic structures using a vegetal metaphor. However, we believe one of the strengths of our modeling framework is its versatility. Only small changes to the modeling process are needed to achieve many different dendritic metaphors.

Figure 3.28: More Dendritic Halftoning

Image	Endpoints	Timing (in secs)
Lena (figure 3.30)	3838	530
Pottery (figure 3.30)	2992	368
Sail Boat (figure 3.30)	3554	546
Cat (figure 3.27)	2019	237
Mandrill (figure 3.27)	2715	446

Figure 3.29: Timing Analysis

3.3.6 Modifying the Modeling Process

Figure 3.31 shows how we can model dendrites that grow upwards from the ground, such as ferns and bushes, with very small changes to the modeling procedure described above. In this case, we set the initial seed point to be at the bottom of the graph, and then we encourage the branches to grow upwards by putting a higher weight on downwards edges. This creates the impression that the dendritic structure is growing upwards like some sort of plant.

There are many other dendritic structures that could be modeled using our approach. Figure 3.32 shows an example of how we can change our modeling step to create structures that resemble stylized lightning. In this case, we chose a point at the top of the graph as the seed and then increase the weight on upwards edges. This coerces the lightning branches to arc downwards, and also ensures that the lightning branches do not curve upwards often. We further promote downwards movement by increasing the number of endpoints towards the lower part of the image.

Figure 3.33 shows random dendritic lightning that has been put through a more realistic rendering process. We can see in this image that our lightning looks reasonably plausible, but remains far easier to control than any means of generating lightning that relies on complex natural processes.

Figures 3.31 and 3.32 show that our modeling process requires only small changes to grow different sorts of structures. The control handles present in our method make it easy to create and control this variety of objects. The rendering of these structures is not the focus of this thesis, and would be almost entirely context-dependent.

It is worthwhile to note that our generic halftoning dendrites can be rendered in a number of different ways to produce compelling results without any changes at all to the modeling process. Figure 3.34 shows some of these results. We rendered our halftoning dendrite first as cracks, such as you might find in pavement, and then as berried vines. The breadth of dendritic structures that can be generated using our method demonstrates its versatility.

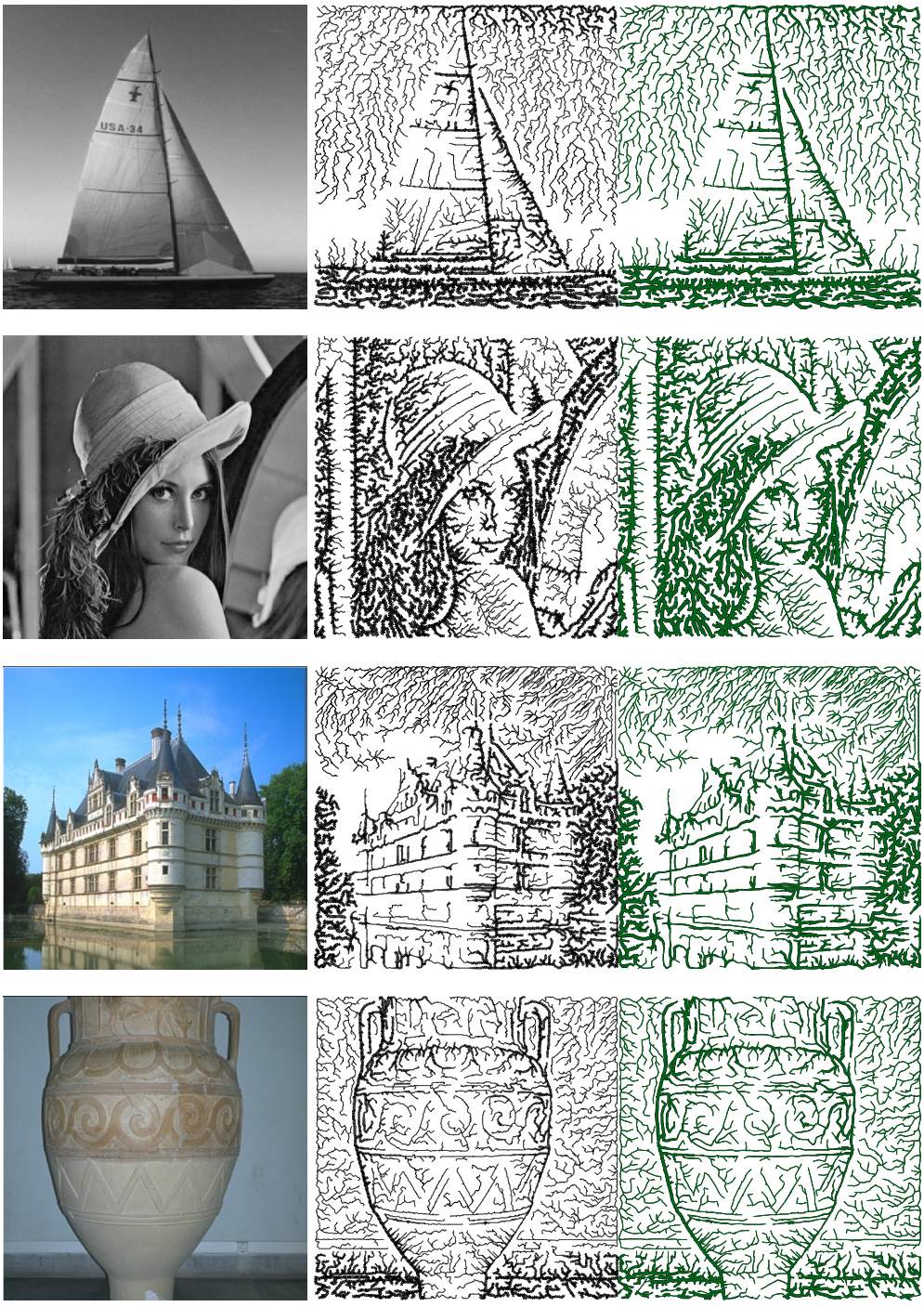

Figure 3.30: More Dendritic Halftoning

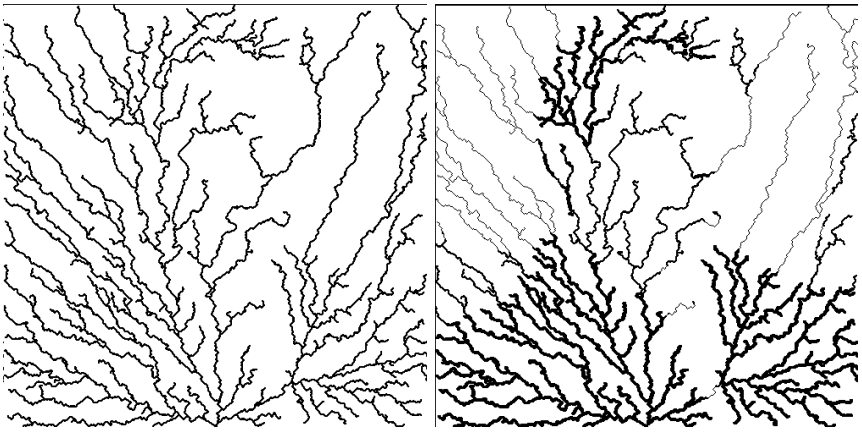

Figure 3.31: Dendrites growing upwards into an image of Eric from figure 3.28 without thickness (left) and with thickness (right).

3.4 Image-Guided Tree Modeling

Pareidolia can be defined as an effect where a natural structure conveys the illusion of a particular image while maintaining a plausible appearance. This effect has been observed in many sorts of natural phenomena. Clouds are an example of objects that are often contended to contain hidden images. We can see another example in figure 3.35 that shows a structure on the surface of Mars that resembles a human face. Figure 3.36 shows that trees and vegetation can sometimes also create pareidolia. In this case, the bent trunk of the tree resembles a bowing person. Most of the more compelling pareidolia tend to involve aspects of human portraiture. People are very adept at recognizing human faces, and as such they are the most common illusion.

Artists have shown some interest in creating pareidolia effects within their work. There is also a precedent for using dendritic structures to convey these hidden images. Tree branches have often been used to form pareidolia, as can be seen in works by Salvador Dali and the poster for the 2007 film 'Premonition'.

These sorts of images require considerable artistic skill to create artificially, as representing the image without compromising the plausibility of the tree branches used to form it is a daunting task. There has been little research into the possibility of procedurally generating these pareidolia effects, likely because of the precision required to achieve these sorts of visual illusions. Our framework gives us the control handles we need to work on this problem.

The first step of this process is to use the framework described in the preceding sections of this chapter to model trees. This requires some small modifications to our general modeling procedure. We also find that using brushfire contours allows us to add interesting foliage to our trees that include high frequency detail as a side effect of the modeling process.

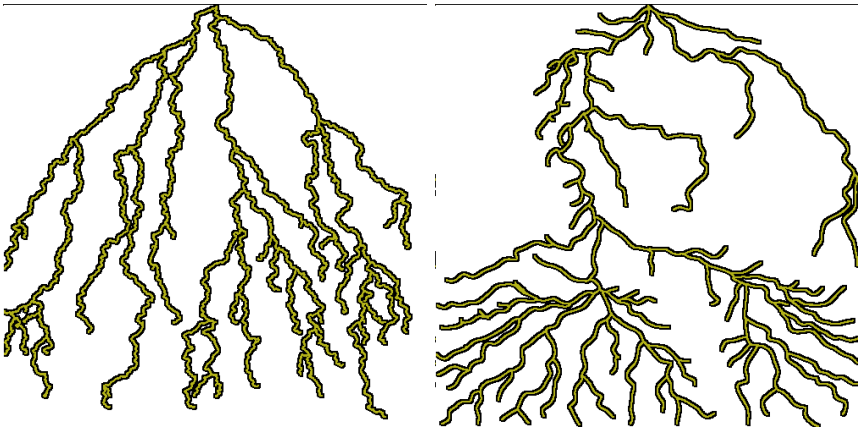

Figure 3.32: Dendrites modeled using a lightning metaphor in a field of uniform random noise (left) and guided by the image of Eric from figure 3.28 (right).

Figure 3.33: A rendering of dendritic lightning (left) and those lightning branches composited into an image of dark clouds (right).

The next step involves combining the image-guided halftoning with the tree structures in order to obtain image-guided trees. This allows us to preserve a tree-like appearance while inserting a portion of the image into the branches. The effect varies depending on the salience of the portion of the image that we capture. Measuring the salience of different regions is an open problem, and we make no attempt to address it within this thesis. Instead, we show that our method can function well assuming appropriate images are used as input.

3.4.1 Growing Trees

The modeling framework that we introduced earlier in this chapter succeeds admirably at representing images in dendritic form. However, we find that several modifications need to be made to our approach in order to effectively model tree structures. The long, winding branches that our

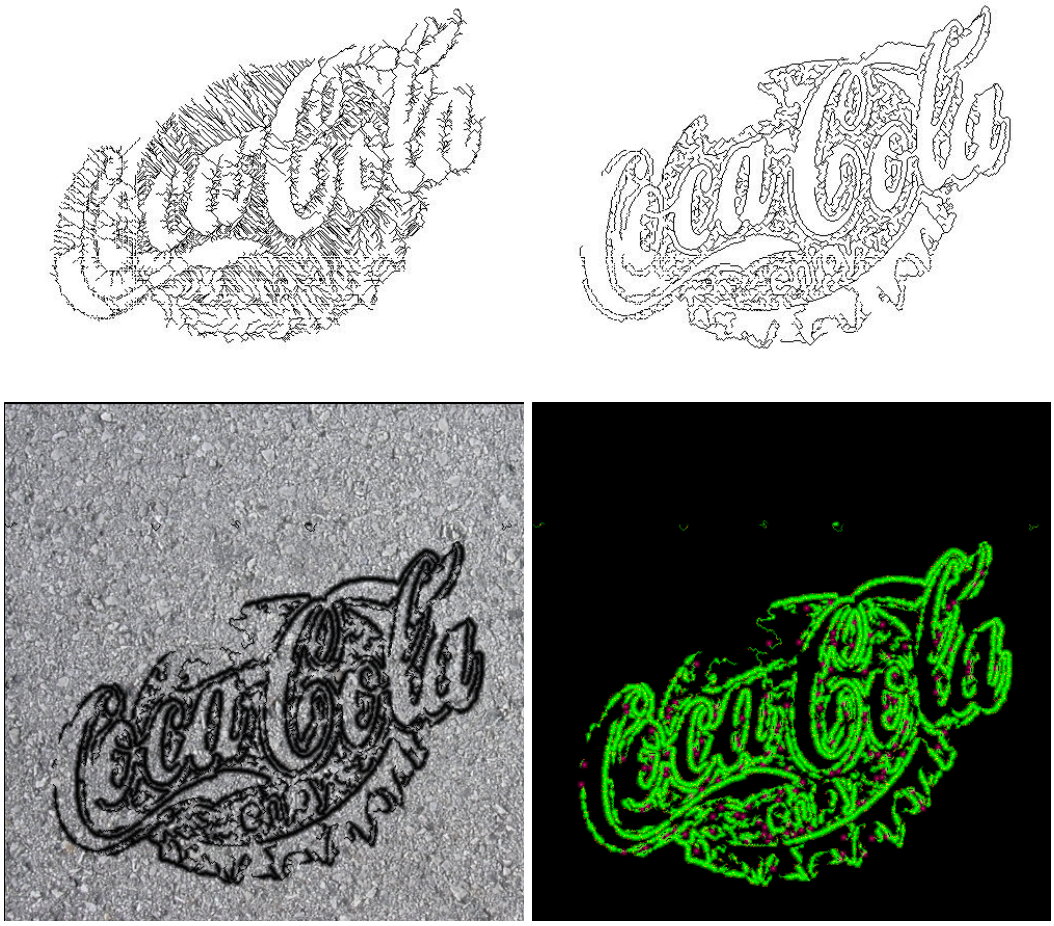

Figure 3.34: Dendritic structures modeled using the Coca-Cola logo as the input image, including halftoning with the cumulative distance metric (top left), the partial non-scalar distance metric (top right) and the halftoning dendrite shown in the top right rendered as cracks (bottom left) and some form of berried vines (bottom right).

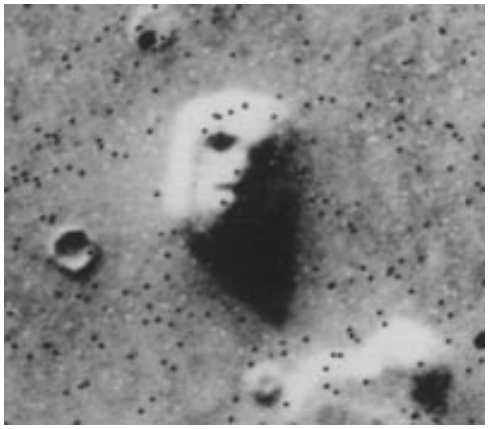

Figure 3.35: A section of the Mars landscape that resembles a human face.

Figure 3.36: A tree that resembles a bowing person.

method produces are not ideal for representing all sorts of tree branches.

Different kinds of trees display different properties with regard to their branches. It is not our intention to capture all these subtle nuances, but instead to generate non-photorealistic trees that look plausibly real. Figure 3.37 shows an example of the kind of painted trees that we are attempting to stylize and model.

The branches that appear on trees can generally be characterized in at least two separate categories. The trunk and primary branches tend to be more straight and regular, while the extremities can sometimes display more winding or gnarled tendencies. The former are more consistent with the paths generated using the cumulative distance metric, while the latter display properties more in tune with our partial non-scalar distance metric. We can accommodate both circumstances by combining the two metrics into a weighted function and changing the contribution of each metric depending on what type of branch we wish to generate.

Figure 3.38 shows a dendritic structure grown over the course of two iterations. Each iteration uses a different combination of the cumulative distance and the partial non-scalar distance metrics,

Figure 3.37: An oak tree.

resulting in different path characteristics. Successive brushfires use every point already on the structure as seeds. In this case, we can see that the branches grow more irregularly on the second iteration (as denoted by the blue branches) which is due to the higher contribution from the partial non-scalar distance metric.

We use this same sort of iterative approach to grow our trees, as shown in figure 3.39. The first iteration uses exclusively the cumulative distance metric, and the seed point is placed at the bottom of the graph. The seed is shown in green in the top left image of figure 3.39. We then generate a sparse distribution of endpoints in order to grow the trunk and the primary branches. In the case shown in figure 3.39, we use only a single endpoint for the trunk. This point should be placed almost directly above the seed in order to preserve the appearance of a tree. Once the endpoints have been placed, we perform our path planning algorithm using the cumulative distance metric to obtain the results shown on the top right of figure 3.39.

Subsequent iterations use the entire existing structure as the seed. This is shown on the bottom left of figure 3.39. We confine a larger distribution of endpoints to a region surrounding the already generated structure. We can find the distance from every prospective endpoint to the structure by running the brushfire algorithm again using every point in the structure as a seed. We can then choose endpoints within the radius using rejection sampling. We have found that putting an upper

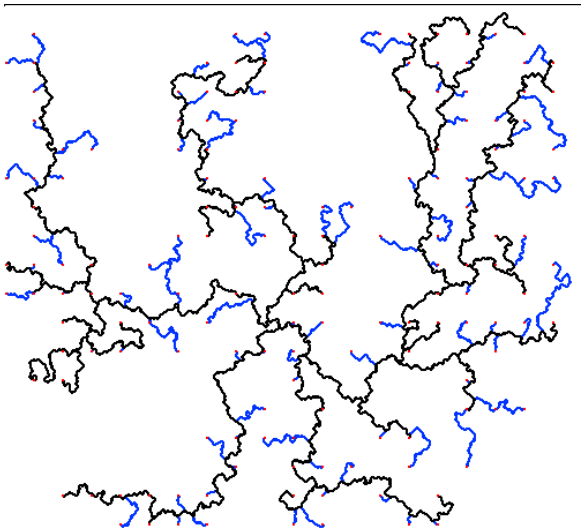

Figure 3.38: A dendritic structure generated through two iterations of our algorithm.

bound on the radius of 120 divided by the number of iterations of the algorithm that have been performed seems to work well. This means that each subsequent iteration will generate shorter branches. Once we have picked all the endpoints, we can path out to them using a metric that progressively approaches the partial non-scalar distance metric to achieve the results shown on the bottom right of figure 3.39.

A tree generated in this fashion is shown in figure 3.40 The branches were rendered using the same splines that were used in our vegetal halftoning, described in the previous section of this chapter. The appearance of our structures resembles the shape of desolate trees that often appear in landscape paintings, or perhaps the proverbial 'haunted' tree. We can see that the cumulative distance metric ensures that the trunk of this tree is fairly regular, while the subsidiary branches are much more twisted due to the increased contribution of the partial non-scalar distance metric.

Once we have established the general structure of our trees, we can consider adding further details that will make them more compelling. One important factor for making the structure of trees more evident is the way the thickness of the branches vary. In stylistic renderings, trees tend to display only a few thickness levels that serve mostly to differentiate the trunk and other primary branches from the rest of the features. We can achieve this effect by adding thickness to all the points in the structure during each iteration of the brushfire algorithm. This means that the trunk and primary branches will remain the most thick, while the periphery will be made up of very thin branches.

Figure 3.41 shows the results of a dendritic tree modeled with iterative branch thickening. There

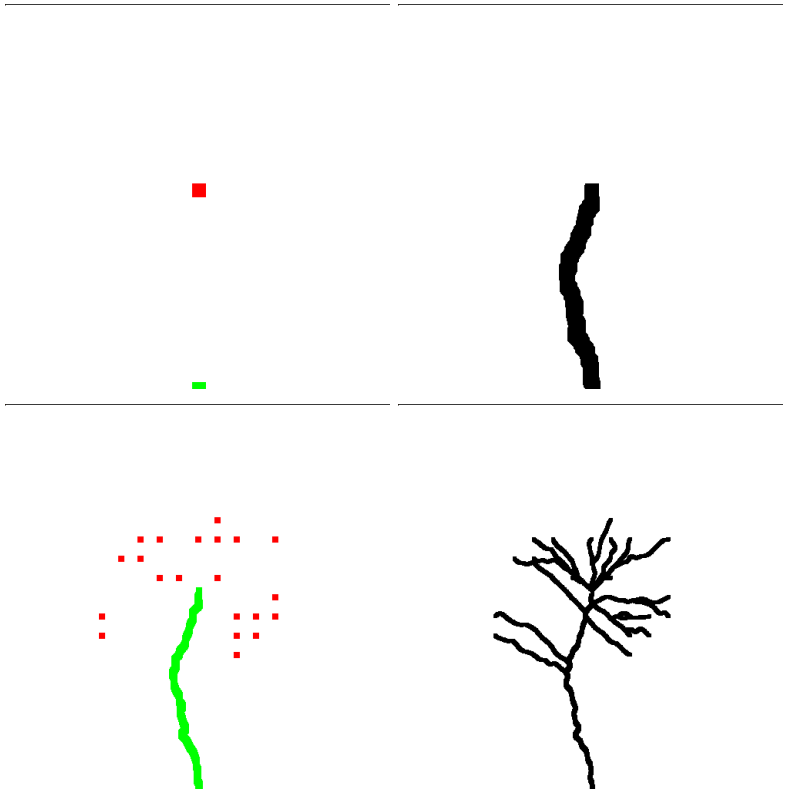

Figure 3.39: The growth of one of our dendritic trees over two iterations. The top images show the first iteration and the bottom images show the second.

are three thickness levels present in this image. The thickness is highest at the trunk, and decreases for each set of sub-branches.

The representation of trees that we have generated thus far works well in some cases, but the two-dimensional nature of our approach does not take self-occlusion into account. Real world trees often feature tangles of branches that occlude one another. We can achieve this effect by applying our algorithm several times to the same trunk structure using different noise maps and then stacking the resulting images on top of one another. This gives the results shown in figures 3.42 and 3.43. We show a more complex stacked tree in figure 3.44, where we created several different trees from two sets of primary branches using different noise maps and then combined the results.

3.4.2 Brushfire Foliage

Our dendritic structures are most conducive to creating skeletal tree branches, but we have found that our approach provides a means of modeling stylized tree foliage as a side effect of our modeling process. In particular, we find that the brushfire contours during various iterations of our algorithm include high frequency detail that resembles certain forms of artistic tree foliage. By applying a

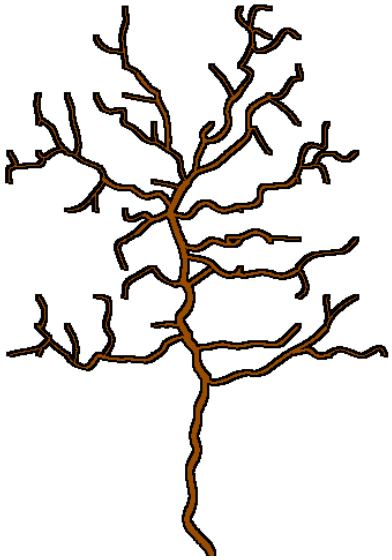

Figure 3.40: A dendritic tree structure built using two iterations of our algorithm.

greenish color map, we are able to achieve interesting results without spending any extra time on our modeling step.

Figure 3.45 shows dendritic trees with our brushfire foliage. The image on the left shows a visualization of the brushfire contours after the first iteration of the tree building algorithm. In this case, the last brushfire was done using the entire trunk of the structure as a seed. As such, the nodes that can be reached more cheaply are clustered close to the trunk, and we index them into a lighter part of the green color map.

The image on the right of figure 3.45 shows the contours using both the trunk and the subsidiary branches as seed points for the algorithm, resulting in foliage clusters that follow the entire tree structure. We artificially prune the foliage on the lower part of the trunk, as we have observed that foliage does not tend to grow directly on the trunk.

The characteristics of the foliage that emerge from this approach are dependent on the distance metric that was used during the last brushfire step. In general, the foliage tends to appear more 'bushy' and irregular when we place more weight on the partial non-scalar distance metric. The contours created using the cumulative distance metric tend to be dominated by proximity to the branches that are being used as seed points. These contrasts were already demonstrated and discussed with regards to figure 3.20 in an earlier part of this chapter.

Figure 3.46 shows a dendritic tree with brushfire foliage applied only during the second iteration of the algorithm. This means that points on the trunk were not considered to be seeds for the second brushfire step. Using only points on the secondary branches as seeds allows us to cover them with brushfire foliage. These contours were generated using a higher weight for the partial non-scalar distance metric, resulting in the high frequency structure along the borders of the contours. We find

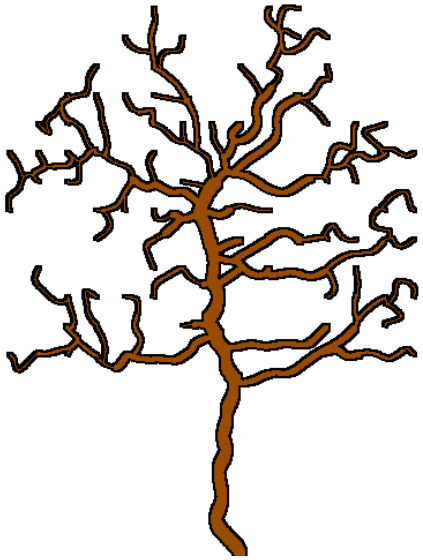

Figure 3.41: A dendritic tree structure generated with decreasing thickness throughout the three iterations of our algorithm..

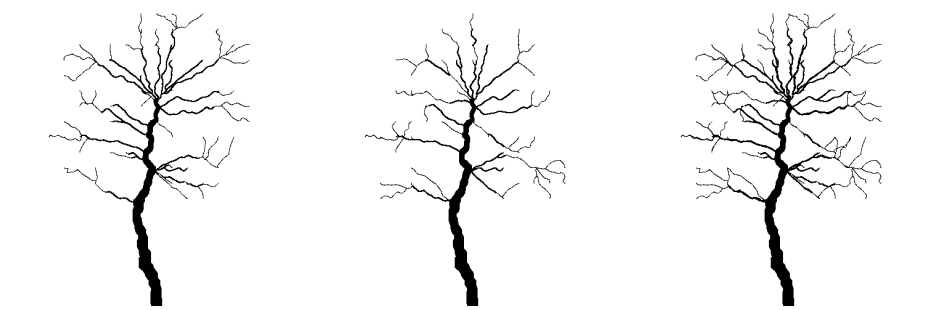

Figure 3.42: By stacking together two dendritic trees (left and center), we are able to create a more detailed tree that includes occlusion (right).

that we can come up with a reasonable method for covering the branches by drawing the brushfire foliage on top if the color value exceeds a certain threshold.

Figure 3.47 shows a dendritic tree structure using brushfire foliage only on the third iteration of the algorithm, which results in it covering only the branches on the periphery of the tree. This effect is similar to what one might see in artistic sketches of trees, where the interior branches are laid bare for examination. The foliage is also generated using more weight on the non-scalar distance metric than any of the previous figures.

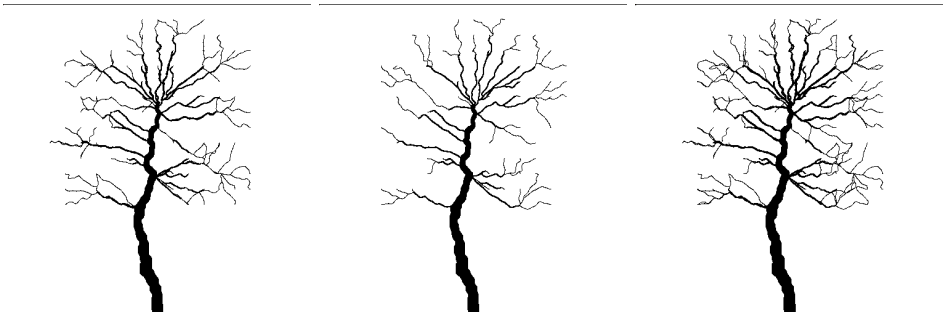

Figure 3.43: A second example of a stacked tree created by combining two trees (left and center) into a single tree (right).

3.4.3 Image-Guided Tree Modeling

Images have a precedent for being used to guide the modeling of various natural structures [33]. In this case, we wish to use some of the principles we discussed with regards to dendritic halftoning in order to plant hidden images within our tree structures. At the same time, we want our trees to maintain a plausible appearance. This is a difficult line to walk, and future work will be required to perfect our approach.

We continue to build our dendritic trees over several iterations, as described above. Our first impulse was to control the evolution of the branches with the image, using the control handles described in our dendritic halftoning application. This allows us to carefully preserve features in the image. However, we found that this led to some very deformed structures that hardly resemble trees at all.

Figure 3.48 shows that simply growing the structure in accordance to an image, even in several iterations of our weighted distance metric, does not create a compelling tree. The branches that emerge from this process tend to be far too regular to suit our needs. However, the shape of the image is greatly diminished without the regularity of certain branches that run along strong gradient edges. In order to accommodate both facets, we elect to embed only a certain portion of the image into our tree structure, and allow the rest to be generated primarily in a random manner, as described in the previous section.

We can see in 3.49 that a portion of Lena's face has been hidden within the branches of the tree structure. This result is generated through several steps, and arbitrarily assumes that the salient features of the image are slightly to the right of its center.

Our algorithm begins by generating the trunk using the cumulative distance metric. The end-point of the trunk is placed in the upper center of the graph, and the edge costs are from a uniform random distribution. Once the trunk has been generated, we build a new lattice using the gradient magnitude and intensity within the input image to determine edge costs, as described with our

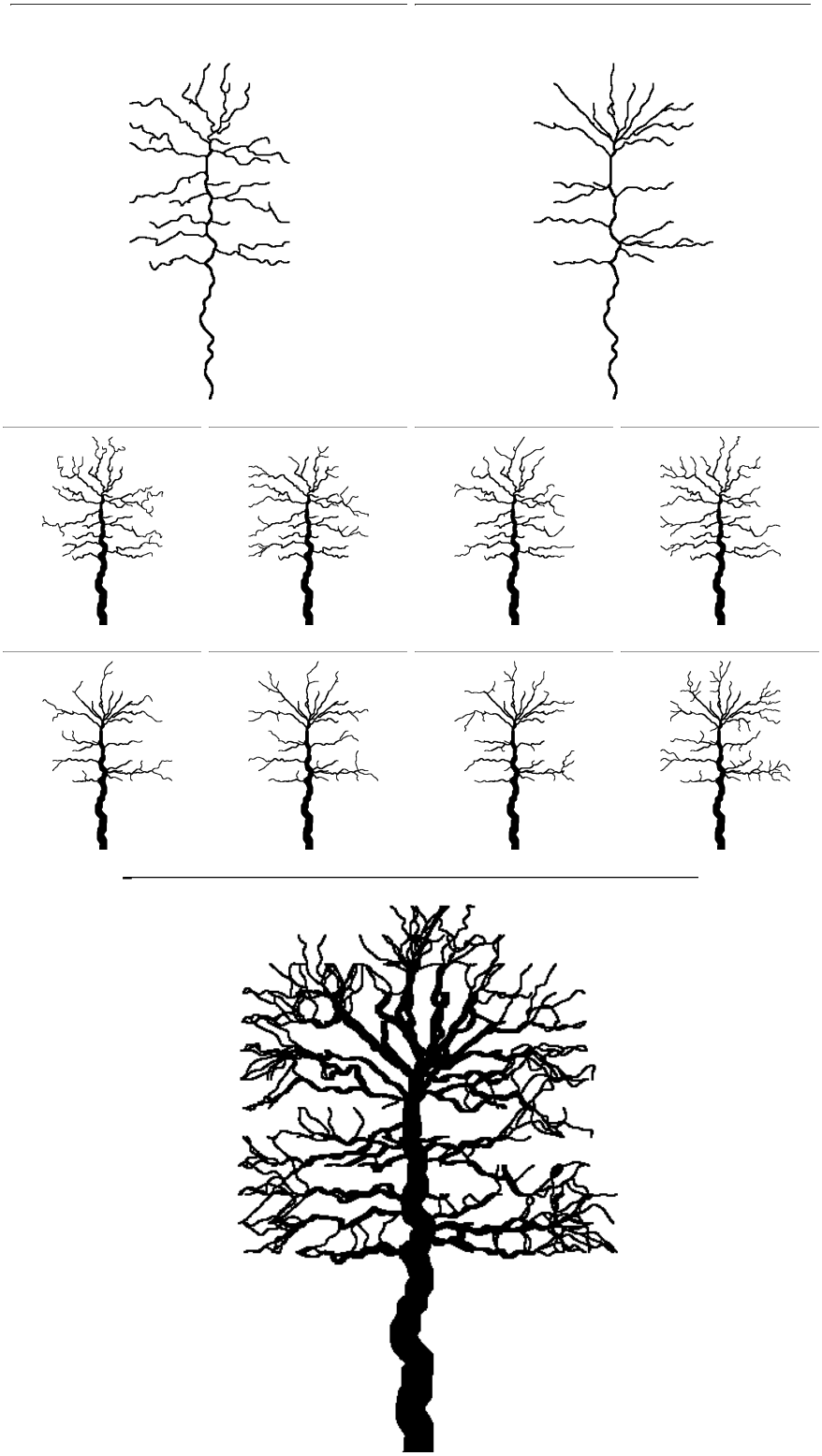

Figure 3.44: An example of a stacked tree where the same trunk was used to create two sets of primary branches (top panels) and then those two were used to generate four sets of subsidiary branches (middle two panels) that were all combined into one final tree (bottom).

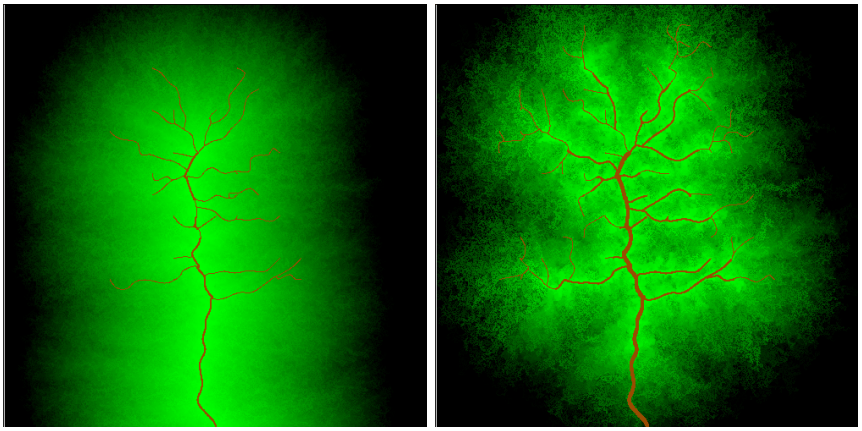

Figure 3.45: Dendritic tree structures using brushfire foliage from the first iteration (left) and the first two iterations (right).

dendritic halftoning process earlier in chapter three. These costs are counter-acted by a random element that increases towards the left side of the image. This means that the paths will follow the image on the right side of the graph, but will include much more noise on the left to help preserve the appearance of a tree.

We then perform a second brushfire step, still with the cumulative distance metric, using the already generated trunk as the seed. In this case, we keep track of the number of edges used to reach each node in the lattice. This gives us the distance from the structure to every node in the lattice, which becomes important when we later wish to limit the area in which the endpoints will be placed to an arbitrary radius around the structure. It is necessary to perform this as a separate step because the partial non-scalar distance metric we use in the next iteration does not give us a good approximation of a radius from the seed due to the winding nature of its paths.

We place endpoints for the dendrite during another brushfire step. The approach we use for endpoint placement has already been described with regards to dendritic halftoning. The only difference in this case is that we only allow endpoints that fall within an arbitrary radius of the current structure. Endpoints that fall outside the chosen radius are still considered seeds for the endpoint selection brushfire, but they are not used as endpoints for the pathing step. For our examples, we decided that only endpoints on the right side of the structure could be chosen in this manner. We then chose approximately half as many endpoints for the left side of the graph, and their coordinates are determined by random rejection sampling. We then introduce a few random endpoints into the right side of the image in order to add some noisy branches into the tree.

Artistic hidden images also use different features within the trees to convey the shape of the image, including the thickness of branches or the foliage. This is demonstrated in the Premonition poster. We would also like to take advantage of this within our implementation. Figure 3.50 shows

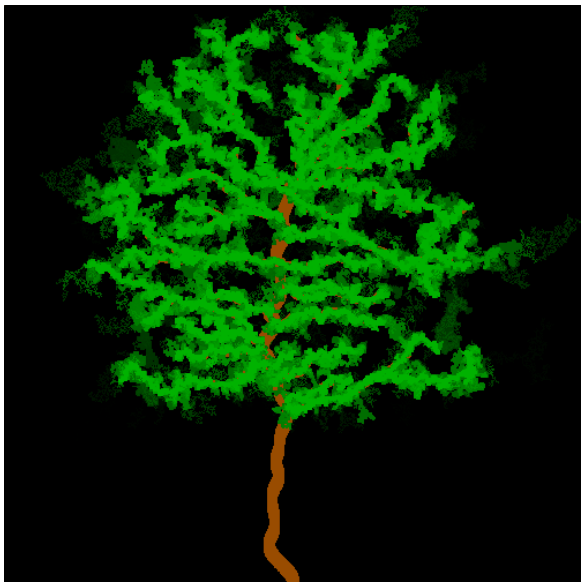

Figure 3.46: A dendritic tree using brushfire foliage during only the second iteration of the algorithm.

an example where the thickness of the branches is based on their importance in the source image, as was done for dendritic halftoning. This approach does a good job of highlighting the image, but it detracts from the plausibility of the tree.

We previously described our iterative approach to tree thickness, which ensures that the trunk and the primary branches are thicker than the periphery. Combining this with importance-based thickness leads to some interesting results. This allows us to both represent the typical branch structure of a stylized tree while maintaining important features in the source image, as shown in figure 3.51.

3.4.4 Tree Rendering

With this modeling process in place, we now turn to the rendering of our image-guided trees. We decided to use image analogies [23] for rendering our results. This allows us to achieve a decent rendering of our trees in a painterly style without spending too much time on writing rendering code, which is outside the focus of this thesis.

Image analogies essentially allow for image filtering from example. This means that the algorithm can learn a transformation function and then apply it to any arbitrary image. In order for this to succeed, a training image must be supplied in two forms, and the algorithm is designed to determine the filter that transforms one form into the other. We can then apply this same filter to our own set of input images in order to transform them into the specific style that was demonstrated by one of the training images. This general approach has proven successful for applications ranging from texture synthesis to painterly rendering. An example of the way we used this algorithm is

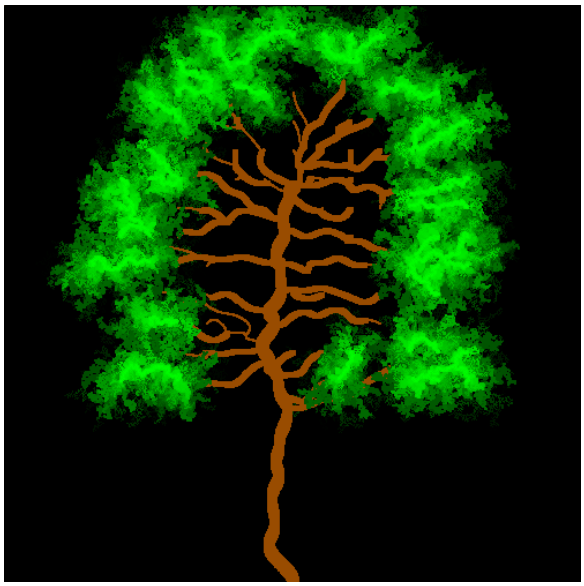

Figure 3.47: A dendritic tree using brushfire foliage during only the third iteration of the algorithm.

shown in figure 3.55.

We recognize that landscape paintings sometimes feature skeletal trees that resemble our output. Thus, we are interested in transforming our results into output that resembles painterly rendering. The first step was to find some landscape images that feature prominent trees. We then transform them into binary form such that the tree itself is distinct from the rest of the image. We can then use this as the original input, and assume that the painted version is the output. This allows the algorithm to learn a filter that transforms the tree into a painted version. This filter is then ready to be applied to our output. In this manner, we are able to produce an output image where the tree is made up of strokes reminiscent in style to the training image.

Figure 3.55 shows the results of using a tree from a painted scene as input to image analogies and then applying that same filter to our trees. First, we apply the results to a basic skeletal dendritic tree. Next, we show how the results of transforming the tree containing Lena's image into a painted form.

We also tried using a white tree against a night sky as our training image. The results from this attempt are shown in figure 3.56. Once again, the same filter was applied to both a basic dendritic tree and to one of our image trees. We use the Lena tree in these examples because Lena remains one of the most recognizable images within the computer graphics community. Figure 3.57 shows a transformation using the same analogy that includes branch and importance thickness in the input image.

Figure 3.61 shows one of the trees that we generated composited into a painted image. A winter landscape painting by American impressionist John Twachtman was chosen to fit with the skeletal

Figure 3.48: Source image (left) and an early dendritic tree containing that image (right).

nature of our dendritic tree. We decided to place our tree far in the background so that a viewer would not immediately be able to discern that the style of our painted tree is different from the one that was used to create the landscape. As shown here, our tree does not look out of place in this background. In fact, some viewers even assumed that this image was showing a real world example of pareidolia.

We use a landscape painting by American portrait painter John Neagle as the background in figure 3.62. We then composited a tree shaped to resemble Eric's face into this backdrop. Once again, we have reduced the scale of our tree in order to diminish the differences in painting style between our tree and the landscape.

It is important to note that our choice to use portraits as the sources for our image-guided trees was very deliberate. Previous work has noted that the human visual system is very adept at recognizing human facial features [20]. This might be one reason why pareidolia that appear in nature often seem to involve faces. We try to take advantage of this same property in burying portrait images in our trees. We have found that the results tend to be the best when the salient features of the portrait are close to the trunks of our trees.

There remains a great deal of future work in the area of hidden images. We have investigated using dendritic trees to subtly convey a source image, but we believe there is potential to embed images in many other natural or stylized phenomena. We further believe that the models generated using our technique could be rendered in a more sophisticated fashion by building a custom rendering engine that is tailor-made for this application. We also believe that more work could be done on finding a balance between branch and importance thickness that would show off the source image without badly damaging the appearance of the tree.

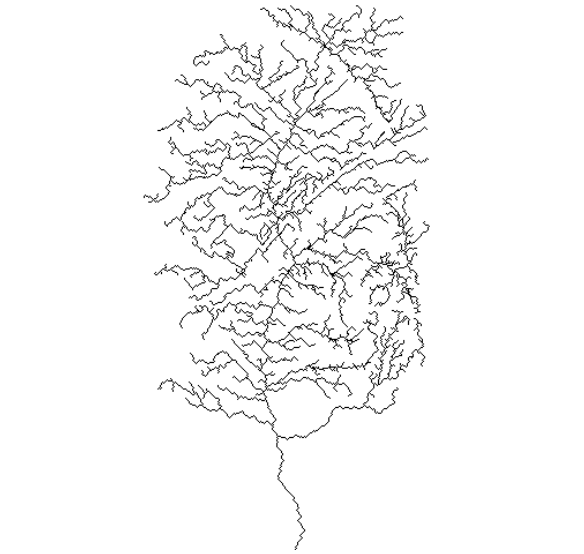

Figure 3.49: A dendritic tree with the ubiquitous Lena image embedded within its branches.

Figure 3.50: A dendritic Lena tree with importance thickness.

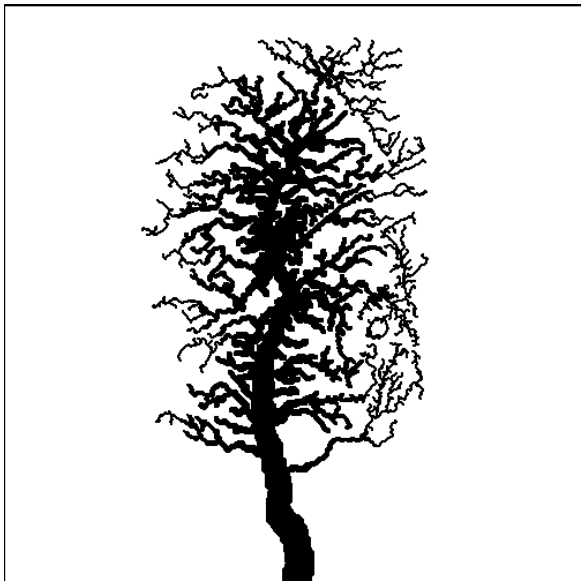

Figure 3.51: A dendritic Lena tree with branch thickness with a small amount of importance thickness.

Figure 3.52: Source image (left) and a dendritic tree containing that image (right).

Figure 3.53: Trees containing images of Eric using importance thickness (left) and branch thickness with a reduced radius (right).

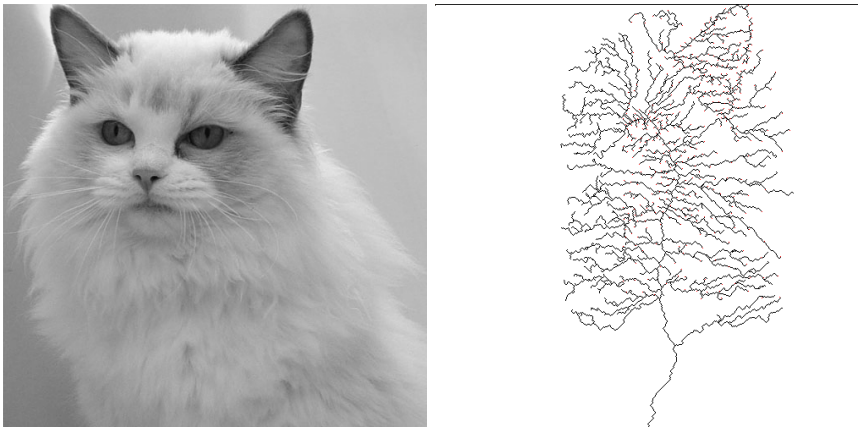

Figure 3.54: Source image (left) and a dendritic tree containing that image (right).

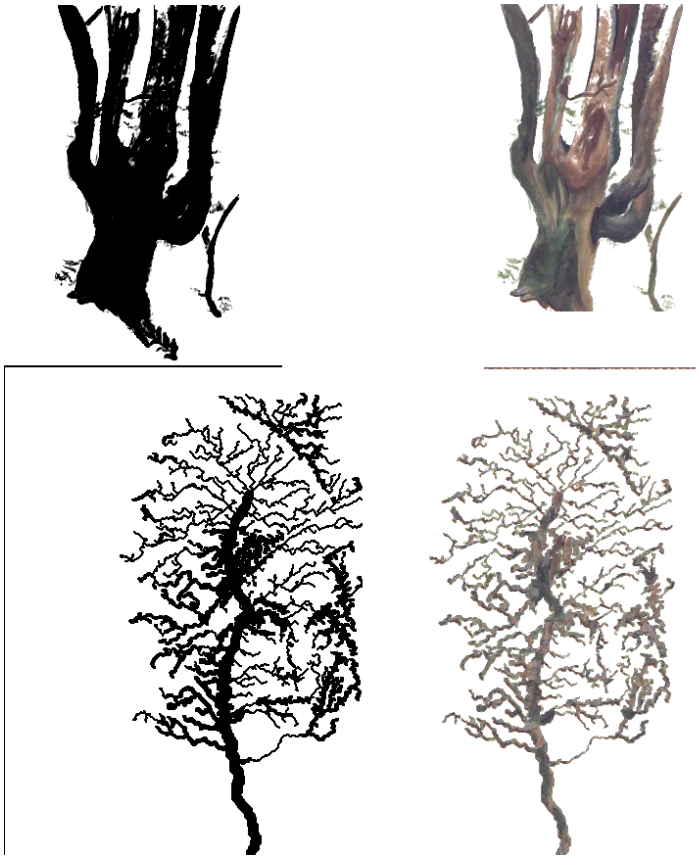

Figure 3.55: Image analogy of a painted tree (top) and the same transformation for a dendritic tree containing an image of Lena (bottom)

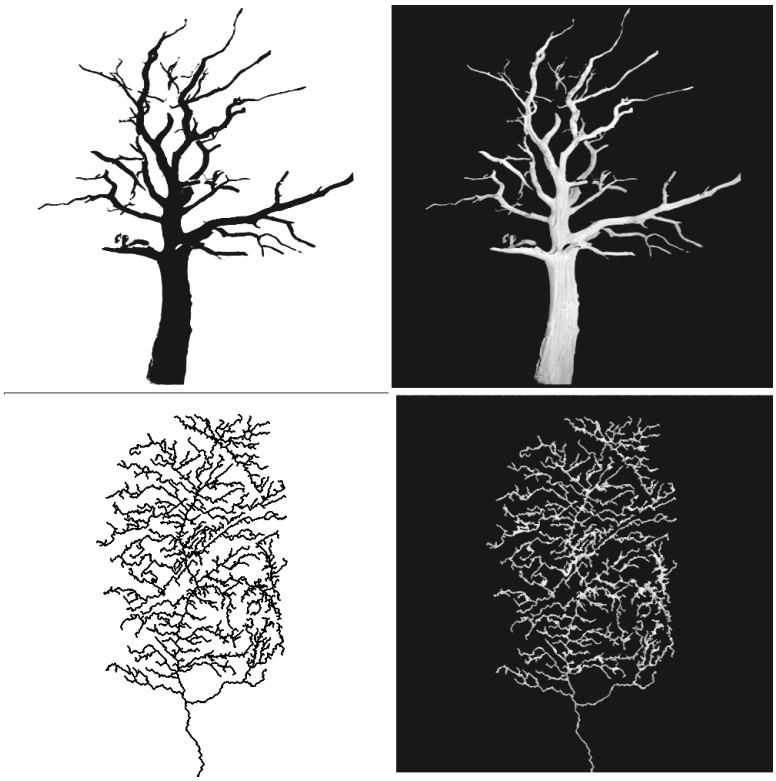

Figure 3.56: Image analogy of a white tree against a night sky (top) and the same transformation for a dendritic tree containing an image of Lena (bottom)

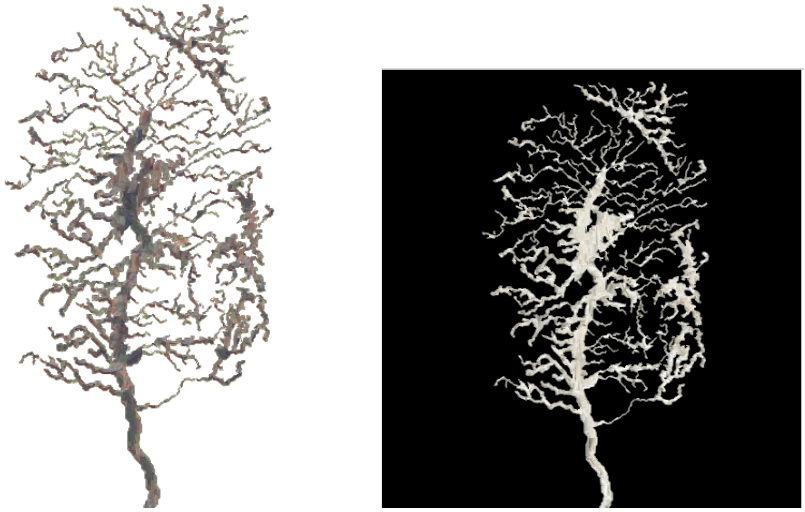

Figure 3.57: The same transformations applied in figures 3.55 and 3.56 performed on a tree containing the image of Lena using branch and importance thickness.

Figure 3.58: The same transformations applied in figures 3.55 and 3.56 performed on a tree containing the image of Eric using branch thickness.

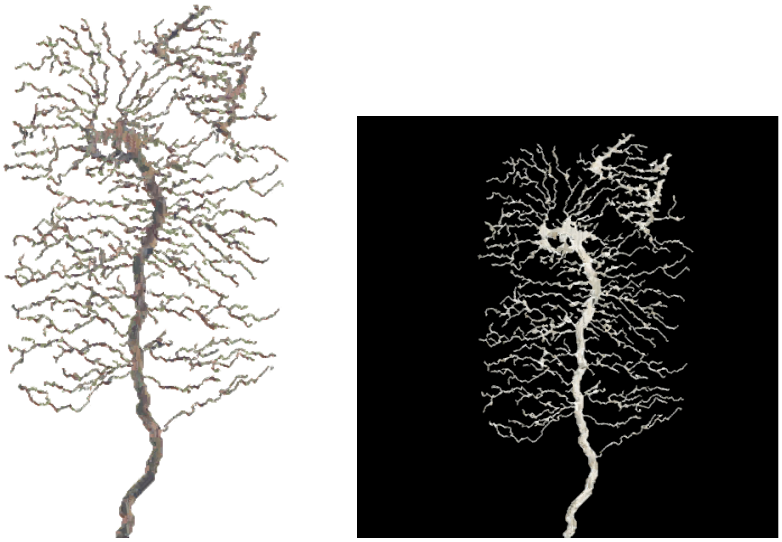

Figure 3.59: The same transformations applied in figures 3.55 and 3.56 performed on a tree containing the image of a cat using branch and importance thickness.

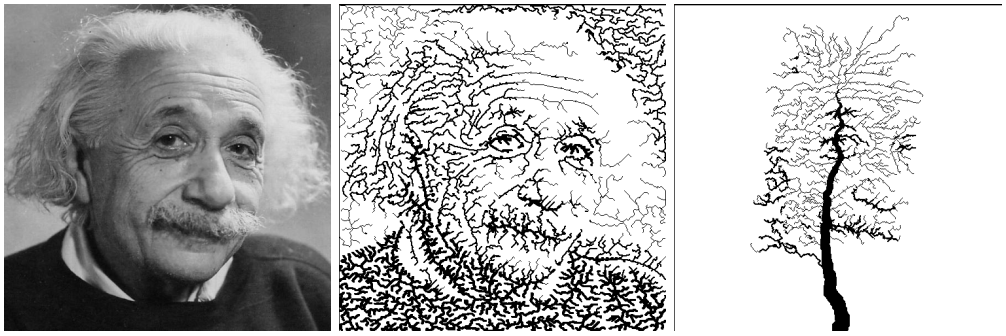

Figure 3.60: Source image (left), halftoning dendrite with thickness (center) and a tree containing the image of Einstein using importance thickness (right).

Figure 3.61: A painted landscape by John Twachtman with one of our trees composited into the background.

Figure 3.62: A painted landscape by John Neagle with one of our trees composited into the background.

Chapter 4

Improved Image Quilting

Texture synthesis from example is the process of taking an input texture chip and using it as a basis for generating an arbitrary quantity of similar texture, without obvious repetition. One way this can be accomplished is by copying parts (pixels or patches) from the input texture chip and pasting them together as an output image.

Though many methods have been developed using a combination of pixel- and patch-based approaches, there remains much active research in this area. In this section, we discuss several methods that are used to remove seams and discontinuities in patch-based synthesis, and propose our own method, which is a modification on the minimum error boundary cut first introduced into texture synthesis by Efros and Freeman [14].

At the end of this section, we compare results obtained using two different methods for shaping the texture patches. We find that our method tends to avoid the most grievous discontinuities, both because it actively decreases the maximum error along the boundary cut and because the winding path that results is more difficult for the human eye to perceive. This principle has already been introduced in the area of lapped textures [41], and it works to our advantage in this context.

4.1 Image Quilting with the Non-Scalar Distance Metric

We briefly describe the process of Image Quilting [14] upon which this work is based. In order to synthesize an output texture of arbitrary size, we divide an input chip into a number of patches. We synthesize the results by copying a patch from the input that best matches the overlap region with the already synthesized portion of the result. In order to shape the patches, we create an error map over the overlap region and find the shortest cost path from the ends of the patch to the other ends. This cut defines which parts of the overlap region will be left as the already synthesized result and which pixels will be considered part of the new patch. An example of the minimum error boundary cut is shown in figure 4.1, with the path through the error map in red.

As described above, our method differs from traditional image quilting in how the shortest path through the error map is determined. Figure 4.2 shows patches shaped using the traditional cost metric against those calculated using our non-scalar method.

Figure 4.1: Error maps and texture patches for image quilting (top), improved image quilting (middle), and a non-scalar metric using 30 maximum edge costs (bottom), patch size of 32 x 32 and an overlap size of 14.

Figure 4.2: Uncut patch (left) and the same patch after the minimum error boundary cut generated with the conventional distance metric (center) and the non-scalar distance metric (right).

The patches that emerge from improved image quilting are more irregular, a characteristic consistent with the metric's attempts to wind around high cost areas. We have noticed that increasing the overlap size tends to greatly improve the quality of results. In our examples, we found that best results could be achieved by using an overlap region roughly equivalent to half the patch size. This is somewhat contrary to regular image quilting, where the cumulative distance paths do not tend to wander far from the routes with fewer edges, meaning that adding greater overlap space does not have much impact upon the quality of results. The increase in quality of the non-scalar metric as the size of the overlap region is increased is shown in figure 4.3.

By closely examining the error profiles of two patches being cut with identical constraints along the overlaps, we can see that the partial non-scalar metric features a noticeably lower peak, as seen on the right of figure 4.4, while the left part of figure 4.4 shows the conventional metric. Winding around the peak in the error map causes the non-scalar metric to generate a longer path, but this

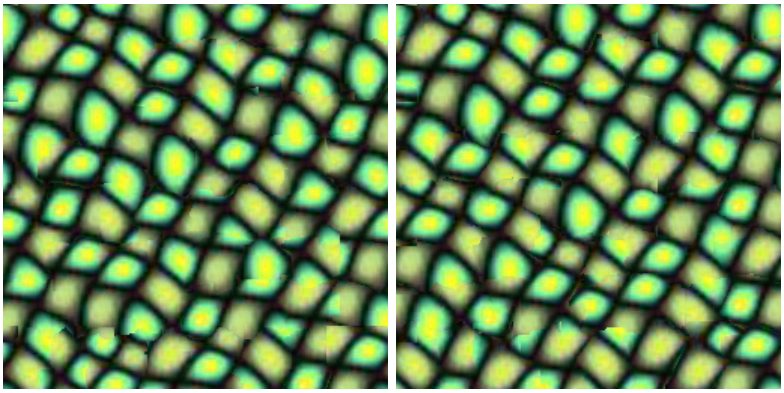

Figure 4.3: Non-scalar metric with an overlap of 10 (left) and an overlap of 14 (right).

allows it to avoid the high error areas that the conventional metric cuts right through. In essence, the non-scalar metric spreads the error out over a long distance, making it less sharp and apparent. In contrast, the traditional distance metric tries to take a short, compact route, even when that means travelling over an error peak.

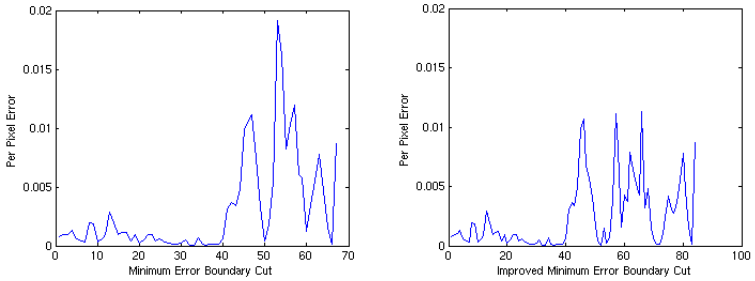

Figure 4.4: Per pixel error profile along the minimum error boundary cut (left) and the partial non-scalar boundary cut (right).

We built our implementation for Improved Image Quilting using the Matlab framework for Hybrid Texture Synthesis [35]. This allowed us to take advantage of several advanced features such as adaptive patch sizes. This technique can be used to split patches into smaller ones when certain user defined error thresholds are exceeded.

The Hybrid Texture Synthesis framework uses feathering by default as a means of shaping patches. We first added functionality for the traditional minimum error boundary cut by creating a heap within Matlab and populating it with paths across the overlap surface. We then added functionality for the partial non-scalar metric [38]. As with feathering, we still believe that pixel re-synthesis can be modified and applied as a post-process pixel in order to reduce errors in our results.

4.2 Improved Image Quilting Results

This section compares results generated using the two different methods for patch shaping described in detail within this paper: the traditional minimum error boundary cut and our improved image quilting method. We decided not to include feathering in this comparison because it does not work as well for the sorts of textures in which image quilting is typically most effective – those where we wish to preserve strong, localized details. Also, feathering is not often used as a standalone process for other contexts in which the minimum error boundary cut is used.

We chose a set of 3 initial textures, and applied each method to them. We selected difficult input textures that have highly localized features because these are the sorts of cases where image quilting is frequently applied. As we show in figure 4.9, our method is still capable of producing compelling results with high frequency input textures.

Figures 4.5, 4.6 and 4.7 show the input texture used and the output generated from conventional image quilting and our algorithm. The initial patch placement was hard-coded, and the best matching patch was chosen at each iteration. This means that the results initially share the same patch sequence, and that they are easily reproducible.

These results demonstrate some of the promising aspects of our approach. An examination of the textures included in this paper shows that we were able to succeed both in reducing the gravity of the most grievous errors that resulted from image quilting, and that we were able to hide seams more successfully by allowing them to take more circuitous routes.

In general, the human eye tends to be drawn quickly towards the most grievous defects in an image. The traditional image quilting method will accept large errors if they serve to maintain a low cumulative distance. Our approach specifically seeks to minimize peak edges, and results in fewer major defects. As a result, casual observation of figure 4.5 is drawn primarily towards the perceptually pronounced errors produced by regular image quilting.

In addition to reducing peak errors, our method allows the minimum error boundary cut to follow a more winding, circuitous path. Although this will not lead to a lower total error than the cumulative distance approach, it will serve to spread the error over a longer distance. Furthermore, we note that the human eye has greater difficulty detecting a seam that follows a more irregular path.

The traditional image quilting textures show defects where the boundary cut has travelled through high cost regions, leaving noticeable defects in its wake. Our non-scalar approach is more likely to wind around such areas, reducing the gravity of the defects produced and making the seams more difficult to detect.

We also attempted some variations on our partial non-scalar distance metric. This involved increasing the number of leading edge costs taken into account when comparing two prospective

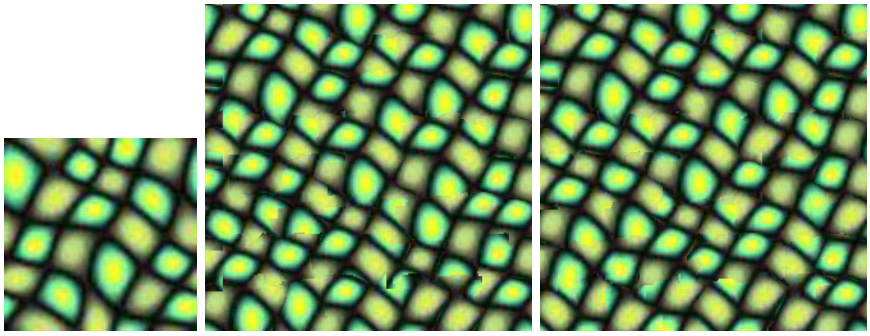

Figure 4.5: Input texture (left), image quilting (center) and improved image quilting (right). Both outputs were generated with a patch size of 32 x 32 and an overlap size of 14.

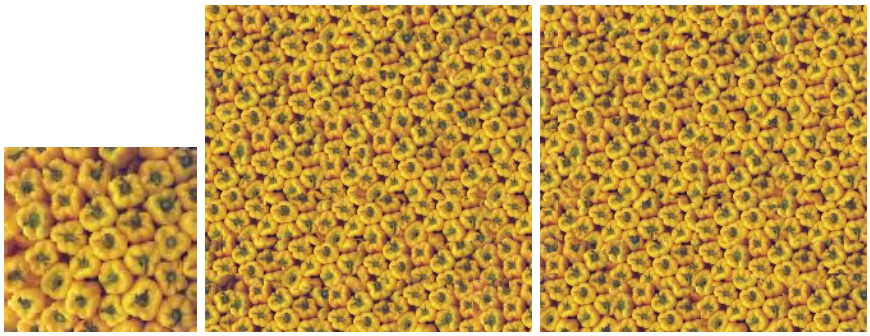

Figure 4.6: Input texture (left), image quilting (center) and improved image quilting (right). Both outputs were generated with a patch size of 32 x 32 and an overlap size of 14.

paths. Figure 4.8 demonstrates that there appear to be diminishing returns as the partial non-scalar metric approaches the full non-scalar metric. Future work in this area might involve a more detailed examination of the trade-off between the quality of results and machine resources as more leading edge costs are employed in the cost comparisons.

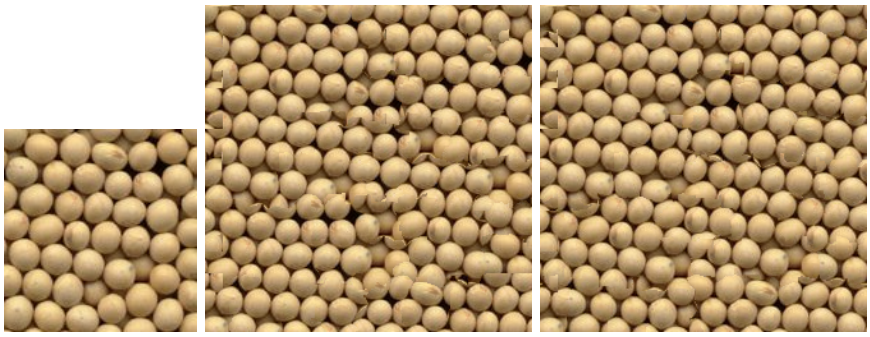

Figure 4.7: Input texture (left), image quilting (center) and improved image quilting (right). Both outputs were generated with a patch size of 32 x 32 and an overlap size of 14.

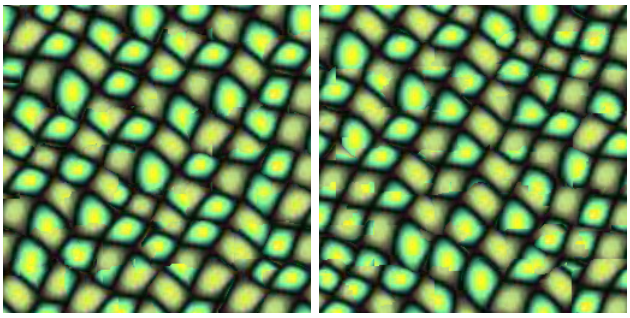

Figure 4.8: Results generated using the partial non-scalar distance metric with 1 maximum edge (left) and a non-scalar metric using 30 maximum edge values (right).

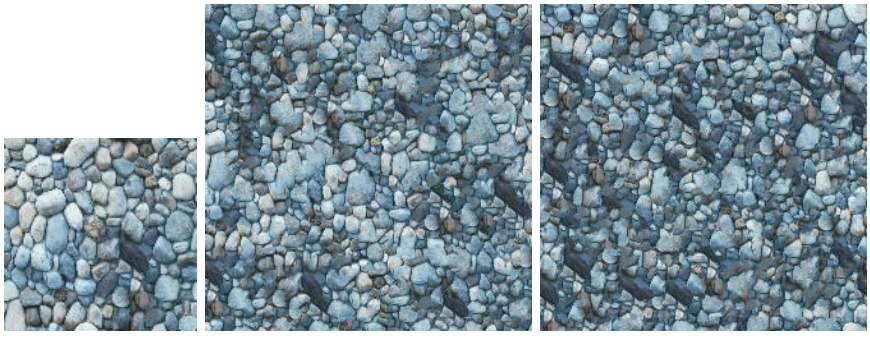

Figure 4.9: Input texture (left), image quilting (center) and improved image quilting (right). Both outputs were generated with an adaptive patch size starting at 32 x 32 and an overlap size of 10.

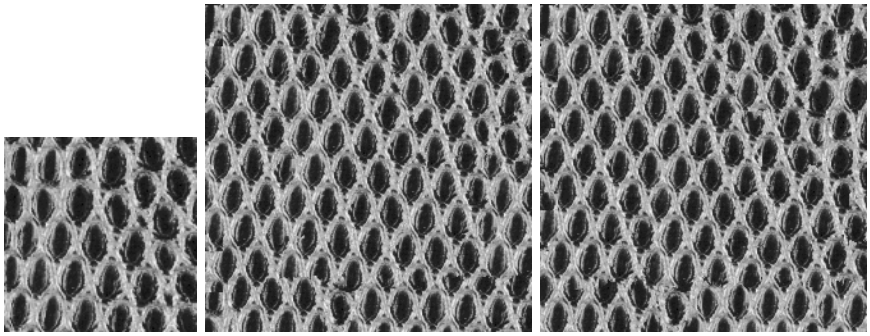

Figure 4.10: Input texture (left), image quilting (center) and improved image quilting (right). Both outputs were generated with a patch size of 32 x 32 and an overlap size of 14.

Figure 4.11: Input texture (left), image quilting (center) and improved image quilting (right). Both outputs were generated with a patch size of 32 x 32 and an overlap size of 14.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Dendritic structures have been used throughout history for decorative and ornamental purposes. Despite their rich history, there has been little success in procedurally generating dendritic structures for artistic effects. Existing procedural methods do not offer the control or versatility necessary to create compelling artistic effects.

The main contribution of this thesis is the introduction of our own procedural method for modeling dendritic structures. Our method uses path planning to achieve superior control and flexibility, and thus avoids the weaknesses exhibited by previous methods. The control characteristics of our approach are further enhanced by the partial non-scalar distance metric that we use, which encourages the branches to obey the control handles more rigorously.

In the first part of this thesis, we surveyed some of the existing procedural methods for modeling dendritic structures and found that they lacked many of the characteristics necessary for creating artistic effects. Next, we presented our own approach for modeling them using path planning and a partial non-scalar distance metric that does not suffer from the weaknesses present in previous methods. In particular, we note that our method provides intuitive control handles to guide the growth of the branches. This exacting control is a prerequisite for creating compelling artistic effects.

The effectiveness of our approach was demonstrated when we applied it to stylistic halftoning and image-guided tree modeling. These applications showed that the partial non-scalar distance metric coerces the paths to follow important features in the input image. We also took a foray into texture synthesis to show that the distance metric we implemented has applications in other contexts.

5.2 Future Work

One avenue for future work would be to look into a wider variety of non-photorealistic metaphors for our dendritic structures. We have already done some work on likening artistic dendrites to

trees and vines, but we believe that our dendrites could be used to model lightning, frost and other branching structures with only minor changes to the modeling process, demonstrating the versatility of our approach. This was shown briefly in figures 3.32 and 3.31 where small changes were made to our modeling process in order to achieve very different dendritic growth. We believe that the main challenge in this regard would be to render the dendritic structures in a manner appropriate to their metaphor.

Another facet we would like to explore in the future would be more complex intertwining behavior. One of the most prominent uses of decorative dendrites throughout history has been the manner in which they interact with nearby features. Figure 3.26 shows that our artistic dendrites are capable of this behavior, but we would like to come up with more complex heuristics in order to simulate highly stylized structures, such as Celtic knots [24].

We also believe that more research could be done with regards to hidden images. Our approach is only intended to deal with the case where the image is being represented within a cluster of tree branches. Artists such as Salvador Dali have hidden images in many other sorts of stylized objects, ranging from the shadows of landscapes to the shapes of clouds. Planting images within these features remains a difficult problem, particularly in the non-dendritic cases. Dealing with these cases might require a completely different approach from the one we used in this thesis.

Furthermore, our approach for dealing with trees yields several avenues for improvement. We would like to extend our implementation to three dimensional trees. Given the inefficiency of path planning for a three dimensional structure, we would like to accomplish this using an artificial means of generating a z value for each point in the tree. The results of this process could be such that the hidden image is only visible from one particular angle, and any shifting of the viewpoint causes other tree branches to obscure the image.

The work on improved image quilting presented in this thesis implies several avenues of future work with regard to texture synthesis. One area we would like to explore would be a better means of determining proper patch placement based on the minimum error boundary cut. Rather than choosing a patch based on the total error present over the entire overlap region, it would seem more fruitful to choose the next patch based on the error characteristics of the lowest cost path through the overlap. This might be very different from the overall error. It seems like this might be especially true for our partial non-scalar distance metric, where the paths can be much longer, but will avoid high cost regions.

We also believe that further research could be done into investigating partial versions of the non-scalar distance metric. Our implementation uses only the maximum edge value when performing calculations, which seems to be enough to produce markedly different results from those generated using the cumulative distance metric, as shown in figure 3.15. In the future, it might be interesting to more thoroughly consider the trade-off between efficiency, memory and the quality of results as

more maximum edge values are stored and the partial non-scalar metric approaches the full one.

It seems likely that our method for generating a minimum error boundary cut based on the partial non-scalar distance metric can be introduced into other contexts where this concept is used, such as Wang Tiles [53, 54, 8] or lapped textures [41]. We believe the characteristics of this method – especially the ability to improve quality using overlap size – are promising, of comparable efficiency to existing methods, and easy to add on top of existing implementations of the minimum error boundary cut.

References

[1] M. Alfonseca and A. Ortega. Representation of fractal curves by means of L-systems. In *APL '96: Proceedings of the conference on Designing the future*, pages 13–21, New York, NY, USA, 1996. ACM Press.

[2] M. Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226, 2001.

[3] P. Ball. *The Self-Made Tapestry: Pattern Formation in Nature*. Oxford University Press, 2001.

[4] W. Baxter, J. Wendt, and M. C. Lin. IMPaSTo: a realistic, interactive model for paint. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 45–148, New York, NY, USA, 2004. ACM Press.

[5] R.M. Brady and R.C. Ball. Fractal growth of copper electrodeposits. *Nature*, 309:225, 1984.

[6] A. Bunde and S. Havlin. *Fractals and disordered systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1995.

[7] N. S. H. Chu and C. Tai. An efficient brush model for physically-based 3D painting. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, page 413, Washington, DC, USA, 2002. IEEE Computer Society.

[8] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. In *ACM Transactions on Grapics (TOG)*, volume 22, pages 287–294, July 2003.

[9] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin. Computer-generated watercolor. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 421–430, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[10] J. Davis. Mosaics of scenes with moving objects. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 354, 1998.

[11] S. Demko, L. Hodges, and B. Naylor. Construction of fractal objects with iterated function systems. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 271–278, New York, NY, USA, 1985. ACM Press.

[12] B. Desbenoit, E. Galin, and S. Akkouche. Simulating and modeling lichen growth. In *Computer Graphics Forum 23*, volume 3, pages 341–350, 2004.

[13] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[14] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer Graphics and Interactive techniques*, pages 341–346, 2001.

[15] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision - Volume 2*, page 1033, 1999.

[16] A. Finkelstein and D. H. Salesin. Multiresolution curves. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 261–268, New York, NY, USA, 1994. ACM Press.

[17] B. Freudenberg, M. Masuch, and T. Strothotte. Real-time halftoning: a primitive for non-photorealistic shading. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 227–232, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[18] L. Gewali, A. Meng, J. S. Mitchell, and S. Ntafos. Path planning in 0/1/ weighted regions with applications. In *SCG '88: Proceedings of the fourth annual symposium on Computational geometry*, pages 266–278, New York, NY, USA, 1988. ACM Press.

[19] B. Gooch, G. Coombe, and P. Shirley. Artistic vision: painterly rendering using computer vision techniques. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 83–ff, New York, NY, USA, 2002. ACM Press.

[20] B. Gooch, E. Reinhard, and A. Gooch. Human facial illustrations: Creation and psychophysical evaluation. *ACM Trans. Graph.*, 23(1):27–44, 2004.

[21] A. L. Guptill. *Rendering in Pen and Ink*. Watson-Guptill Publications, New York, 1976.

[22] M. Harada, A. Witkin, and D. Baraff. Interactive physically-based manipulation of discrete/continuous models. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 199–208, New York, NY, USA, 1995. ACM Press.

[23] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and Interactive techniques*, pages 327–340, 2001.

[24] M. Kaplan and E. Cohen. Computer generated Celtic design. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 9–19, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[25] T. Kim, M. Henson, and M. C. Lin. A hybrid algorithm for modeling ice formation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 305–314, New York, NY, USA, 2004. ACM Press.

[26] T. Kim and M. C. Lin. Visual simulation of ice crystal growth. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 86–97, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[27] T. Kim and M. C. Lin. Physically based animation and rendering of lightning. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04)*, pages 267–275, Washington, DC, USA, 2004. IEEE Computer Society.

[28] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 277–286, New York, NY, USA, 2003. ACM Press.

[29] L. Liang, C. Liu, Y. Xu, B. Guo, and H. Shum. Real-time texture synthesis by patch-based sampling. In *ACM Transactions on Grapics (TOG)*, volume 20, pages 127–150, July 2001.

[30] J. Long and D. Mould. Improved image quilting. In *GI '07: Proceedings of Graphics Interface 2007*, pages 257–264, New York, NY, USA, 2007. ACM Press.

[31] R. Mech and P. Prusinkiewicz. Visual models of plants interacting with their environment. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 397–410, New York, NY, USA, 1996. ACM Press.

[32] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 191–198, New York, NY, USA, 1995. ACM Press.

[33] D. Mould. Image-guided fracture. In *GI '05: Proceedings of the 2005 conference on Graphics interface*, pages 219–226, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[34] D. Mould. Stipple placement using distance in a weighted graph. In *Proceedings of the Computational Aesthetics in Graphics, Visualization and Imaging*, pages 44–52, 2007.

[35] A. Nealen and M. Alexa. Hybrid texture synthesis. In *Proceedings of the 14th Eurographics workshop on Rendering*, pages 97–105, 2003.

[36] A. Nealen and M. Alexa. Fast and high quality overlap repair for patch-based texture synthesis. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, pages 582–585, Washington, DC, USA, 2004. IEEE Computer Society.

[37] S. C. Olsen and B. A. Maxwell. Fluid simulation as a tool for painterly rendering. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Posters*, page 24, New York, NY, USA, 2004. ACM Press.

[38] D. K. Pai and L. M. Reissell. Multiresolution rough terrain motion planning. *IEEE Transactions on Robotics and Automation*, 14(1):19–33, February 1998.

[39] H. Pedersen and K. Singh. Organic labyrinths and mazes. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 79–86, New York, NY, USA, 2006. ACM Press.

[40] J. L. Power, A. J. Bernheim Brush, P. Prusinkiewicz, and D. H. Salesin. Interactive arrangement of botanical L-system models. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 175–182, New York, NY, USA, 1999. ACM Press.

[41] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470, 2000.

[42] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, page 581, New York, NY, USA, 2001. ACM Press.

[43] P. Prusinkiewicz, M. James, and R. Mech. Synthetic topiary. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 351–358, New York, NY, USA, 1994. ACM Press.

[44] P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[45] P. Prusinkiewicz, L. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 289–300, New York, NY, USA, 2001. ACM Press.

[46] W. T. Reeves and R. Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 313–322, New York, NY, USA, 1985. ACM Press.

[47] M. P. Salisbury, S. E. Anderson, R. Barzel, and D. H. Salesin. Interactive pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 101–108, New York, NY, USA, 1994. ACM Press.

[48] A. Secord. Weighted Voronoi stippling. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 37–43, New York, NY, USA, 2002. ACM Press.

[49] M. Shiraishi and Y. Yamaguchi. An algorithm for automatic painterly rendering based on local source image approximation. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 53–58, New York, NY, USA, 2000. ACM Press.

[50] C. Stark. An invasion percolation model of drainage network evolution. *Nature*, 362:329, 1991.

[51] L. Streit and J. Buchanan. Importance driven halftoning. *Computer Graphics Forum*, 17(3):207–217, 1998.

[52] O. Veryovka and J. Buchanan. Halftoning with image-based dither screens. In *Proceedings of the 1999 conference on Graphics interface '99*, pages 167–174, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[53] H. Wang. Proving theorems by pattern recognition II. *Bell Systems Technical Journal*, pages 1–42, 1961.

[54] H. Wang. Games, logic and computers. *Scientific American*, pages 98–106, November 1965.

[55] J. Weber and J. Penn. Creation and rendering of realistic trees. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 119–128, New York, NY, USA, 1995. ACM Press.

[56] L. Wei and M. Levoy. Fast texture synthesis using tree-structured quantization. In *Proceedings of the 27th annual conference on Computer graphics and Interactive techniques*, pages 479–488, 2000.

[57] G. Winkenbach and D. H. Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 91–100, New York, NY, USA, 1994. ACM Press.

[58] P. H. Winston. *Artificial intelligence (3rd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1992.

[59] T. Witten and L. Sander. Diffusion-limited aggregation, a kinetic critical phenomenon. *Physical Review Letters*, 47(19):1400–1403, 1981.

[60] M. T. Wong, D. E. Zongker, and D. H. Salesin. Computer-generated floral ornament. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 423–434, New York, NY, USA, 1998. ACM Press.

[61] J. Xu and C. Kaplan. Image-guided maze construction. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 29, New York, NY, USA, 2007. ACM Press.

[62] Y. Xu, B. Guo, and H. Shum. Chaos mosaics: Fast and memory efficient texture synthesis. Tech Report MSR-TR-2000-32, Microsoft Research, 2000.

[63] S. Zelinka and M. Garland. Jump map-based interactive texture synthesis. *ACM Trans. Graph.*, 23(4):930–962, 2004.

[64] Y. Zhang and R. E. Webber. Space diffusion: an improved parallel halftoning technique using space-filling curves. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 305–312, New York, NY, USA, 1993. ACM Press.

[65] J. Zhao and N. I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Trans. Graph.*, 13(4):313–336, 1994.